REDUCING CLIENT-SERVER COMMUNICATION FOR EFFICIENT REAL-TIME WEB APPLICATIONS

Hatem M. AZIZ

PhD

UNIVERSITY OF BRADFORD

2019

Reducing Client-Server Communication for Efficient Real-Time Web Applications

The Use of Adaptive Polling as A Case Study for Multi-User Web Applications

Hatem Mansour AZIZ

Submitted for the degree of
Doctor of Philosophy

School of Electrical Engineering and Computer Science

University of Bradford

2019

Abstract

Hatem Mansour Aziz

Reducing Client-server Communication for Efficient Real-time Web Applications:
The Use of Adaptive Polling as a Case Study for Multi-user Web Applications.

Keywords: Real-time web, Bandwidth, AJAX, Polling, Web applications, Multi-user.

A key challenge of current multi-user web applications is to provide users with interesting events and information in real-time. This research reviews the most common real-time web techniques to identify drawbacks while exploring solutions to improve simplicity, efficiency, and compatibility within a client-server environment. Two solutions are proposed for enhancing the efficiency of real-time web techniques by reducing client-server communication. First, a model of browser monitoring control observes the browser activity and decides if to postpone client-server communication in the case of inactive tabs. This model was implemented and tested with results demonstrating that a significant number of client-server connections can be avoided in the browser background. These results suggest the solution can be optimised for any real-time technique as it benefits from being a developer side technique that works consistently on all browsers. Second, 'Adaptive Polling' is a pull-based real-time web technique to overcome bandwidth issues of the reverse AJAX method of 'Polling' by controlling the frequency of requesting updates from the

server based on the last server response. This approach is implemented and tested with results showing how a significant number of redundant connections can be avoided while the server does not return updates. This solution is a good alternative to other real-time web techniques as it features low latency, the simplicity of implementation, and compatibility with all browsers and servers.

Declaration

I hereby declare that this thesis has been genuinely carried out by myself and has not been used in any previous application for a degree. The invaluable participation of others in this thesis has been acknowledged where appropriate.

Hatem Aziz

Dedication

This thesis is dedicated to the loving memory of my Mother, to my Father, my Wife, my little daughter and all my family and friends, whose love, help, support, and prayers have made me able to reach this stage and achieve one of my key goals.

Acknowledgments

In the Name of Allah, the Most Gracious, the Most Merciful.

All Praise is Due to Allah for His Glorious Ability and Great Power, who has given me the power, patience and knowledge to complete this doctoral thesis.

I would like to express my gratitude to all those who gave me the support and encouragement to complete this thesis. Specifically, I would like to sincerely thank my supervisor *Mr. Mick Ridley* for his guidance patience and support throughout this research, he has offered me all possible assistance to complete this thesis. Also a special thank you goes to my second supervisor *Dr. Andrea Cullen* for her support and guidance during my research.

I would like heartily to thank my father, my stepmother, my brothers and sisters for all their support and encouragement during my research.

I would like to express very special thanks to my wife for her support, encouragement and love that has given me the strength to complete this thesis.

Many thanks to all my friends for their support to reach this stage of success.

List of Abbreviations

AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
ASF	Apache Software Foundation
ASP	Active Server Pages
CGI	Common Gateway Interface
CSS	Cascading Style Sheets
DB	Database
DOM	Data Object Model
DWR	Direct Web Remoting
GPL	General Public License
HTML	Hypertext Markup Language
JSON	JavaScript Object Notation
JSP	Java Server Pages
PHP	PHP: Hypertext Pre-processor
RDBM	Relational Database Management
RDBMS	Relational Database Management System
RIA	Rich Internet Applications
SGML	Standard Generalised Markup Language
SQL	Structured Query Language
TCR	Temporal Coherence Requirement
TTR	Time To Refresh
W3C	The World Wide Web Consortium

WWW	World Wide Web
XHR	XML Http Request
XHTML	Extensible HyperText Markup Language
XML	Extensible Markup Language
XPath	XML Path Language
XSL	Extensible Stylesheet Language
XSLT	Extensible Stylesheet Language Transformations

Table of Contents

Table of Figures

Table of Listings

Chapter 1

1 Introduction

1.1 Introduction

Today, we live in the era of information with a significant amount of new data added to the web every second. According to (Gunelius 2014), every minute there are nearly 2.5 million Facebook posts shared, 72 hours of video uploaded to YouTube, and over 200 million emails sent. This explosion of data over the Internet has changed the culture of serving web users where people can select the type of data and information they want to receive without the need for repetitive direct searching on the web. Therefore, web applications today are focused on providing information that is in the interest of their users. In other words, users can subscribe to the exact information they want to know about and receive an email or notification providing access to that relevant information. The time factor for delivering subscribed information is also crucial as users want to be notified of the latest updates as they occur requiring web applications to respond in real-time. This type of service called real-time client-server communication, which is adopted by most popular websites and social media applications, such as Facebook and Twitter.

Multiple real-time web techniques have been introduced to deal with providing users with updates as they occur at the server. These techniques follow a variety of concepts for delivering such a service and are based on the type of communication between the client and server, including **client-pull, server-**

push and **dual-channel** communication that define the Polling, Comet, and WebSockets technique, respectively. These approaches can all achieve real-time communication between the client and server, but issues exist that affect their suitability and satisfaction for an implementation. Factors that must be considered when selecting which technique to apply include scalability, bandwidth, latency, compatibility, security, and efficiency.

1.2 Research Scope and Terminology

1.2.1 Research Scope

This thesis studies the most common real-time web techniques to understand their mechanisms, advantages, problems, and the possibility of improvements in terms of simplicity, compatibility, and efficiency. The challenge is how to improve these techniques and propose a generic solution as a reasonable alternative to the current real-time web technologies.

1.2.2 Research Terminology

- **Real-time web**: the characteristic of a web application that enables users to receive notifications of updates happening at the server without explicitly search for each occurrence.
- **Pull**: a real-time web concept of frequently sending requests from client to server and receiving updates if available.
- **Push**: a real-time web concept of triggering the server by the client to send an update of an interest once available.
- **Polling**: a real-time web technique based on the Pull concept.
- **Compatibility**: the property of a web technology that can work on most browsers and servers without special features or adjustments.

- **Simplicity**: the ease of implementing websites and web applications based on real-time web technologies.
- **Efficiency**: the ability to accomplish requirements with high quality and the best possible use of available resources, i.e., the least time and effort (“What is efficiency” 2018). In terms of the real-time web, efficiency can be defined by a set of website characteristics that reflect efficiency, such as:
    - Ease of implementation.
    - Responsiveness and interactivity.
    - Fast loading of pre-designed content and desired updates.
    - Works on most browsers and servers.
    - Less server bandwidth.
    - Less waste of resources, such as CPU, network, data usage, and power consumption (e.g., batteries for some devices)

1.3 Motivations

Web applications have changed dramatically over time evolving from static web pages designed to display information on a user’s machine over the Internet to produce dynamic and interactive websites where the user plays a key role in customising the interface and retrieving desired information. Websites, such as Facebook and Twitter, provide users with real-time information (as they occur). This type of feature is commonly described in the subscriber or observer design pattern where users or agents are notified about changes at a central server or from other users as they happen.

This technological leap has introduced many languages, tools, and techniques to provide effective and real-time web applications. AJAX (Asynchronous JavaScript And XML, see 2.4.1) is one of the most popular tools for enabling Rich Internet Applications (RIA), which are web applications that appear and function as traditional desktop applications (Moore et al., 2007). Along with its ability to enhancing the efficiency and interactivity of web applications, AJAX technology also supports real-time web applications through several techniques as will be described in 2.4.1.

The motivation of this research is summarised through three ideas. First, we understand the general concept of the real-time web and its most used technologies, such as AJAX. Second, we study the differences between these real-time web techniques. Third, we highlight the weaknesses of these techniques, if any, with observations for improvement. Fourth, we suggest an alternative real-time web technique with consideration to efficiency and simplicity for situations involving multi-user updates, such as auctions and hotel booking websites.

1.4 Aim of the Research

This research aims to produce an alternative real-time web technique by identifying an efficient solution with fewer drawbacks and increased simplicity of implementation. Our solution is expected to be well suited for a variety of scenarios of real-time web applications.

1.5 Objectives

The following objectives are established to fulfil the research aim.

- To investigate the most common real-time web techniques and understand their differences as well as their advantages and disadvantages.
- To propose a browser monitored real-time web technique to control client-server communication for reducing unnecessary connections.
- To implement and test the proposed browser monitoring solution with a real-time web technique.
- To introduce a modified model of Polling as a reverse AJAX real-time technique by controlling the interval of client-server requests according to the frequency of server updates.
- To produce a prototype of the modified Polling to demonstrate the new mechanism and implement the design through a simple web page that communicates with the server.
- To test and validate the proposed new mechanism for Polling.

1.6 Contributions

This research studies real-time web technologies and the possibility of enhancing these techniques by creating an alternative to suit multiple cases based on real-time web communication. The following highlights the contributions provided through this research.

- A model for enhancing the efficiency of real-time client-server communications by monitoring user browsing activities in a way that reduces the number of connections when no active browsing for the targeted web page is occurring (see 4.3). This model is implemented

through the real-time technique of Polling with a chat web page. This solution is tested with a simple simulation to compare the original and optimised techniques. The results show this model saves a large number of unnecessary connections between the client and server, which reduces the bandwidth and the use of server resources. This work was published in Aziz and Ridley (2017).

- An enhanced model of a real-time web technique (Polling) to eliminate its disadvantages of bandwidth and redundant server connections by monitoring the frequency of pushed server updates to the client, which modifies the interval needed to send the next server request. This change can involve decreasing or increasing the interval according to if the update frequency is fast or slow (see 5.3).
- Implementing 'Adaptive Polling' based on the proposed model to improve the efficiency of Polling and overcome its significant drawbacks (see 5.4 – 5.7). A prototype is designed and implemented as a conversation web page using AJAX technology, PHP, MySQL, and JavaScript. In this implementation, we monitor the rate of updates pushed by the server to control the interval of the next client-server connection by either increasing or decreasing the interval within a specified limit (e.g., minimum and maximum). This solution is tested through a simulation of the Polling and Adaptive Polling techniques with the participation of several students over the local network. The test case results show that we can reduce the number of connections to one-sixth using our approach of Adaptive Polling compared to the original method of Polling. Based on these positive results, this

enhanced technique can be adopted as an alternative real-time web technique. This work was published in Aziz and Ridley (2016).

1.7 Thesis Structure

The chapters critically review the underlying technology, theory, and current advancement of real-time web-based technological solutions.

Chapter 2 overviews the tools, databases, and web technologies needed for building dynamic and responsive websites and web applications with a focus on reviewing the fundamental real-time web techniques in terms of their mechanisms, features, and drawbacks to establish a foundation for possible improvements.

Chapter 3 critically reviews previous related work that focuses on developing solutions for resolving real-time web techniques' problems or enhancing performance and efficiency to highlight the need for more development and improvements in these techniques.

Chapter 4 presents an enhanced model for improving the efficiency of client-server communication in real-time web techniques based on monitoring user movements (at the browser) to reduce communication with the server to reduce server bandwidth and unnecessary resource use. This model is implemented and tested with the results explained to express the importance of this work.

Chapter 5 introduces the concept of 'Adaptive Polling' as an enhanced real-time web technique based on the reverse AJAX technique of 'Polling' to improve efficiency and reduce the bandwidth of client-server communications.

Application scenarios are produced followed by a prototype and its implementation tested against the basic technique. A comparison with previous work is also explained to reflect the differences and appreciate the value of the proposed technique.

Chapter 6 summarises the contributions of this research and presents the conclusion as well as further improvements and recommendations for future work.

Chapter 2

2 Background and Fundamentals of Real-time Web Technologies

2.1 Introduction

This chapter focuses on the most common technologies used to build real-time websites and web applications. Beginning with an overview of database and web technologies required to build a typical website, it advances to the techniques and tools offering the functionality to produce more responsive and interactive websites with fast-loading and smooth update refreshes. The AJAX approach and its mechanisms and functionalities will be reviewed for how they enable the desired features of modern websites. Finally, its vital role in building real-time web applications will be highlighted along with a broader review of the most common real-time web techniques in terms of their mechanisms, advantages, drawbacks, and suitability for implementation.

2.2 Database Technologies

A database is a collection of related data stored in such a way as to be maintained, manipulated, and retrieved quickly when needed. Where a Database Management System (DBMS) is a set of software and applications designed to manage database processes such as storing, maintaining, and retrieving data. The job of a DBMS is typically combined with other considerations such as security and integrity of the data stored in a structured database (Sumathi and Esakkirajan 2007).

2.2.1 Relational Database

The relational database is the most common database model that is used by developers for building dynamic websites to serve information over the Internet. As reviewed by (Sumathi and Esakkirajan 2007), in this model, data objects are represented as relations, which adhere to rules for managing the following objects and attributes:

- Tables of information, called “relations.”
- Rows of information, each called a “tuple,” that contains all the data of a single instance of a table.
- Attributes label columns of a table.
- An essential part of the design process includes selecting “attributes” of objects, which can be stored together in a table without redundancy.
- The “key” for a relation is a set of attributes with values that uniquely define the tuple.
- Indexes are data structures that assist in retrieving or changing information in tables.

2.2.2 Relational Database Management Systems

A Relational Database Management System (RDBMS) is a database system based on the relational model standards of SQL (see 2.2.3) to add, edit, and retrieve data from a database. There are common RDBMS, such as Oracle and Microsoft SQL Server. MySQL and PostgreSQL are additional systems that are open source (Sharanam and Vaishali 2008).

2.2.3 SQL

SQL is a Structured Query Language used by an RDBMS to manipulate data. As outlined by (W3Schools 2017), the following commands perform the core operations of SQL on a database:

SELECT - extracts data from a database.
UPDATE - updates data.
DELETE - deletes data.
INSERT INTO - inserts new data.
CREATE - creates a new table/database
ALTER – modifies table/database

2.2.4 MySQL

MySQL is an open-source relational database management system licensed under a General Public License (GPL). MySQL is widely used in web projects as it can deal with a large amount of data for storing, manipulating, and retrieving with ease. MySQL can also process different types of data from a single character to large files and graphics. It provides a fast, reliable, and easy-to-use server, which has encouraged many developers to rely on it for building websites. MySQL was developed and provided by MySQL AB, a commercial company that provides services around the MySQL database. It was subsequently owned by Sun Microsystems, which was then taken over by Oracle Corporation. (Widenius and Axmark 2002)

2.3 Web Technologies

2.3.1 HTML

HyperText Markup Language (HTML) is based on SGML (Standard Generalised Markup Language) to provide a framework for generating markup

language to create web pages that deliver types of data such as text, images, and multimedia files over the Internet. Through the use of HTTP (HyperText Transport Protocol), HTML uses tags to format elements with most including start and end tags (Mercer and Shannon 2003), for example:

<title> … </title>

HTML has evolved through several versions. The classic HTML 2.0 supported the core HTML elements and features, such as tables and forms, but there existed a lack of consideration for advanced features, such as scripting and frames. HTML 3.0 and HTML 3.2 improved on the previous version by supporting presentation elements such as font. With HTML 4.0, the focus was to provide the basis for the transition to Cascading Style Sheets (CSS) and later to remove most of the presentation elements from the HTML specification, including font, to encourage formatting using the CSS approach. Also, it supported framed documents that were not considered in the previous versions (Powell 2010).

2.3.2 HTML5

HTML5 resulted from a cooperative effort between W3C (the World Wide Web Consortium) and WHATWG (the Web Hypertext Application Technology Working Group). While work on HTML5 remains in progress and not all features are supported by all browsers, developers and web designers welcome the advanced features available through HTML5 (MacDonald 2011). This latest version aims to avoid the strictness of XHTML (discussed below), and makes significant improvements to traditional HTML by introducing new elements to handle document layout, such as <header>, <footer>, <article>, and <nav>. New elements are now included for embedded multimedia, such

as <video> and <audio>. HTML5 also introduces the concept of WebSockets as a real-time web technique, which is considered in this research for building efficient real-time web applications.

2.3.3 XHTML

XHTML stands for eXtensible Hypertext Markup Language and was announced after the success of the popular XML (eXtensible Markup Language), which is utilized in many aspects of programming. XHTML is similar to HTML 4.01, but stricter as its markup language must follow XML rules. This requirement offers many benefits to XHTML, such as all tools designed to work with XML can be also used by XHTML (Duckett 2008). The key differences between XHTML and HTML are the following:

- A DOCTYPE declaration must be included to indicate the version of XHTML. For example:

  ```
  <!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
  ```

- The XML namespace attribute must be included in the <html> tag. For example:

  ```
  <html xmlns="http://www.w3.org/1999/xhtml">
  ```

- Elements must be properly nested and closed.
- Elements must be in lowercase.
- Attribute values must be quoted.

2.3.4 CSS

Cascading Style Sheet (CSS) is a language designed to improve the layout of HTML documents and enhance their appearance. This framework uses a set of rules to control the style of HTML elements on a single web page or

throughout an entire website. CSS is supported by the W3C, the organisation which also defines the standards of HTML (Shelly et al. 2002). The styles included in CSS may be implemented in three ways:

- Inline: included within the HTML element as a tag's property.
- Embedded: written in HTML documents to control one or more of an element's appearance.
- Linked: styles for elements are saved in separate files with the extension .css and referenced by the HTML document through a link.

2.3.5 CSS3

CSS3 is widely used for modern websites and is an update to previous versions and remains compatible with all CSS versions. CSS2 added features of positioning, and CSS3 introduced new features and properties to enhance the HTML presentation and save effort and time for developers to build higher quality web pages (Stone 2016). These new features include:

- Adding rounded corners to borders and images.
- Supporting animations and text effects.
- Defining text columns to organize data presentation.

2.3.6 Client-side Scripting

The purpose of client-side languages is to manipulate, control, and manage the content of web pages. This class of scripting provides high interactivity between the user and browser while ensuring a swift response from the server when processing client requests. JavaScript, for instance, is one of the most common scripting languages as it is user-friendly and has broad support from libraries (Sarkar 2016).

2.3.6.1 JavaScript

JavaScript appeared as a result of a collaborative effort in 1995 between Netscape Communications Corporation and Sun Microsystems, Inc. JavaScript is an object-based scripting language that works on the client side through the web browser to make web pages more dynamic (Pollock 2010). JavaScript can build dynamic web pages, display alert boxes, open new browser windows, validate information in forms, and perform calculations (Keogh 2005).

JavaScript may be directly incorporated within the tag <body> of the HTML document by writing the code between a starting tag <script> and closing tag </script> as in Listing 2-1.

```
<html>
<head>
      <title> Using JavaScript </title>
</head>

<body>
      <script language="Javascript" type="text/javascript">
      document.write('Hello, this is from JavaScript');
      </script>
</body>
</html>
```

Listing 2-1. A JavaScript example.

2.3.6.2 jQuery

jQuery is a library of JavaScript built to simplify scripting within HTML documents by using the (DOM) Document Object Model, which allows access to the elements of HTML for manipulating their values to change the appearance, position, and events. Moreover, jQuery functionalities can be triggered on the occurrence of events such as webpage load, text change, and mouse hover (Beighley 2010).

2.3.6.3 VBScript

Visual Basic Scripting (VBScript) is a scripting language from Microsoft under the framework of the Visual Basic programming language. With a syntax similar to Visual Basic, VBScript was introduced to work within a web environment and, specifically, the client side. It offers similar functionality to JavaScript as it can be embedded in an HTML document to perform various tasks by accessing the DOM of that document. VBScript may have cross-browser compatibility issues because it is not natively supported by some browsers, including Firefox and Opera. VBScript can be embedded within the ASP language (see 2.3.7.4) to execute processes on the server side. Listing 2-2 demonstrates an example of VBScript (Kingsley-Hughes et al. 2004).

```
Dim datBirth
datBirth = InputBox("Please enter the date on which " & _ "you were
born."
If IsDate(datBirth) Then
      datBirth = CDate(datBirth)
      MsgBox "You were born on day " & Day(datBirth) & _
               " of month " & Month(datBirth) & " in the year " & _
               Year(datBirth) & "."
```

Listing 2-2. VBScript example.

2.3.7 Server-side Technologies

A variety of technologies are used on the server-side to process client requests and communicate with the server to produce requested information and push it back to the browser. The scripting languages PHP, JSP, and ASP are included in this class of technology and play a key role in building dynamic and interactive websites.

2.3.7.1 CGI

The Common Gateway Interface (CGI) specifies the standards of transferring information between the web server and the browser. A CGI program can be written in a wide variety of languages such as Perl and C. Historically, Perl was the most popular and suitable for writing CGI scripts, and its interpreter is preinstalled on most existing servers. CGI scripts are easy to implement, offer a quick response, and are compatible with most servers (Hanegan 2001). However, a problem with CGI is that each time a CGI script is executed, a new process is started, which may cause a slow down on the server for busy websites. A server API, such as ISAPI (Internet Server Application Program Interface) or NSAPI (Netscape Server Application Programming Interface), offers a possible solution although it faces other difficulties with implementation. Another alternative growing in popularity is the Java servlet (Beal 2013).

2.3.7.2 Perl

Perl is a high-level programming language that was first designed for text processing because of its effective built-in functions and is now considered a general purpose programming language (Chromatic 2014). For Perl support, a group of people introduced CPAN, a website for the Comprehensive Perl Archive Network, that includes thousands of modules for download and installation to reduce the code and save development time. Also, Perl is a good language due to its text handling capabilities, which is an essential characteristic for web development. However, many web hosting providers do not provide an up-to-date version of Perl, which minimises the benefits offered from its latest features (Pepersack 2016).

2.3.7.3 PHP

PHP originally stood for “Personal Home Page” and was renamed to “PHP: Hypertext Preprocessor.” PHP is a general-purpose scripting language released as a free open source project in 1997 to meet the needs of creating web pages that can efficiently process data. However, PHP alone is not enough to build dynamic websites as it requires a server that can process its scripts. Apache is a free web server that can be installed on a local machine to allow developers to test PHP scripts locally. Also, PHP integrates with relational database management systems, such as MySQL and PostgreSQL, to produce efficient websites that meet users’ needs (Lengstorf 2009).

There are many features included with PHP. First, it can be embedded into HTML tags by inserting code between a beginning tag “<?php” and ending tag “?>.” PHP can be run on both UNIX and Windows servers making it more accessible compared to ASP (see 2.3.7.4). Moreover, PHP is suitable for building large and complex web applications because it allows for advanced programming and is easy to integrate with web pages and MySQL (McLaughlin 2012).

2.3.7.4 ASP

Active Server Pages (ASP) is a technology that can be used to design a powerful and dynamic web applications at ease, benefiting from being supported by Microsoft, ASP files use HTML tags and VBScript as a default scripting language but also can use JavaScript. (Korol 2008) Different versions of ASP exist including ASP 1.0, ASP 2.0, and ASP 3.0, and the name of the latest versions changed to ASP.NET as it leverages the .NET framework. This evolution of ASP allows it to be written with different languages, including

VB.NET and C#, as they can be executed in the Common Language Runtime (CLR) environment (Evjen 2006).

2.3.7.5 JSP

Java Server Pages (JSP) is a Java technology that allows developers to create portable and dynamic web applications. JSP can construct custom actions allowing a single line in a JSP page to provide considerable functionality. Similar to PHP, JSP integrates with HTML so that its code may be embedded within HTML tags to create dynamic web pages. This feature uses Scriptlets designated by placing valid Java code within the tags “<%” and “%>.” JSP is another powerful language for building complex web application that works Apache Tomcat, which is an open source servlet container developed by the Apache Software Foundation (ASF) (Brunner 2003).

2.4 AJAX for Interactive Web Technologies

2.4.1 AJAX

AJAX initially stood for Asynchronous JavaScript And XML, but its concepts can now be implemented by using additional tools, such as ActionScript, which integrate with Flash platforms, and JSON for transferring data in different formats (Gross 2006). AJAX is a combination of existing web technologies for providing better integration of web services where it can be used for different purposes, such as autocompletion, the load on demand, and drop-down list-boxes based on other inputs (Segue 2013).

In traditional communications between a client and server, as shown in Figure 2-1 (Allen 2009), the entire page must be refreshed when receiving data from the server. This process involves extended loading times and redundant data

usage. In an AJAX approach, transferring data between the server and the client occurs in the background so that the browser does not need to wait until all data is received to refresh the page. In other words, with AJAX there is no need for the browser to refresh the entire page, but only the portions that have an update. Web applications that employ this approach through AJAX are referred to as “asynchronous” (Holzner 2007).

The AJAX concept is very useful for building modern websites that are based on real-time notification systems and real-time server updates where only subsets of a webpage are modified upon a change of data at the server. Thus, this background client-server communication approach enables fast-loading and responsive websites with smooth displays of updates. Figure 2-2 (“Introduction to Ajax” 2016) illustrates asynchronous communication where the AJAX part is represented at the client-side (e.g., browser), which controls passing requests to the server and processes the returned data to display it to the user in its final form. Additional details about AJAX mechanisms and its implementation is covered in 2.4.3.

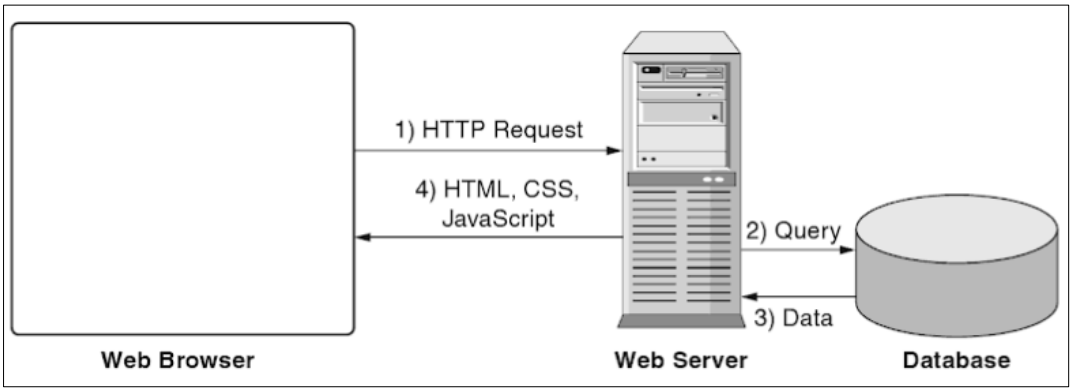

Figure 2-1. Traditional web application communication.

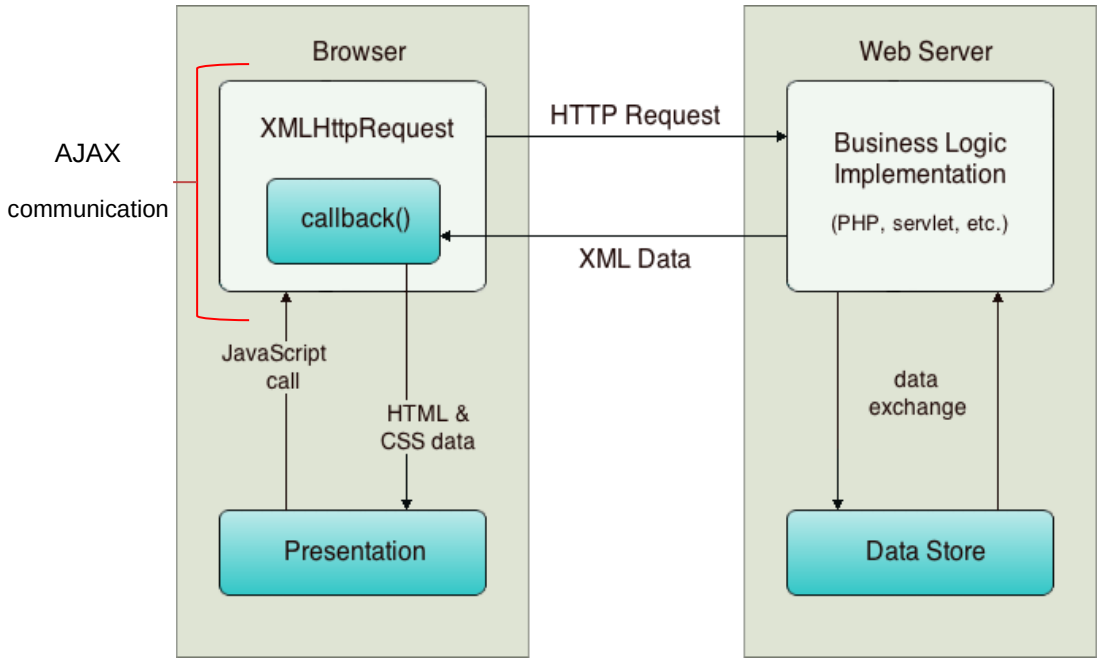

Figure 2-2. Asynchronous communication (AJAX-style).

2.4.2 AJAX Tools

AJAX combines multiple technologies that integrate to produce asynchronous and interactive web applications (Larsen 2011), and are listed in the following:

- Standards-based presentation using XHTML and CSS.
- Dynamic display and interaction using the DOM.
- Data interchange and manipulation using XML and XSLT.
- Asynchronous server communication using XMLHttpRequest.
- JavaScript for binding everything together.

2.4.2.1 DOM

The Document Object Model (DOM) is a platform- and language-independent interface designed to provide high interactivity with an HTML document by allowing access and dynamic updates to the document. The DOM works effectively with JavaScript to produce interactive web applications (Hégaret

2017). In addition, the DOM is subject to differences between browsers, so AJAX must handle these differences when using the DOM to obtain the same results across all browsers. Listing 2-3 provides an example of how JavaScript uses the DOM to update the object *msg* and to change its background colour in an HTML Document.

```
var Msg="Example 1";
document.getElementById('msg').value = Msg;
document.getElementById('msg').style.backgroundColor='orange';
```

Listing 2-3. An JavaScript example using the DOM.

2.4.2.2 XML

The eXtensible Markup Language (XML) uses embedded symbols (called tags) within the text to describe the structure of a document and to organise data such that it can facilitate retrieving it dynamically in various formats including plain text, databases, and comma-delimitated spreadsheets. Also, the same information can be quickly sent to and presented on different devices from mobile phones to PC web browsers (Dykes and Tittel 2005).

According to (Gulipalli 2015), the flexibility and ease of using XML have made it widespread today. XML is not restricted to predefined tags or commands but features a collection of rules that must be considered allowing programmers to create custom tags and structures as long as they follow the general framework.

In an XML document, data is saved as a collection of elements structured in such a way as to be related so that they can be retrieved and exchanged between compatible and incompatible applications. This feature is important

for the reuse of stored data from different applications (Kahate 2009). In Listing 2-4, a simple code snippet shows a possible XML format for a car object.

```
<car>
<manufacturer>BMW</manufacturer>
<type>Saloon</type>
<model>5 Series</model>
<colour>White</colour>
<year>2012</year>
</car>
```

Listing 2-4. Simple code illustrating the XML format and structure.

2.4.2.3 XSLT

The Extensible Stylesheet Language for Transformations (XSLT) is designed to transform XML documents into different display formats, such as HTML, Portable Document Format (PDF), and flat text file, as shown in Figure 2-3 (Tidwell 2001). XSLT also uses the XPath language to search for and recognise elements and attributes within XML documents. Therefore, the XSLT processor (engine) works efficiently according to a predefined template to build the final presentation of the original XML document (Wikipedia 2016).

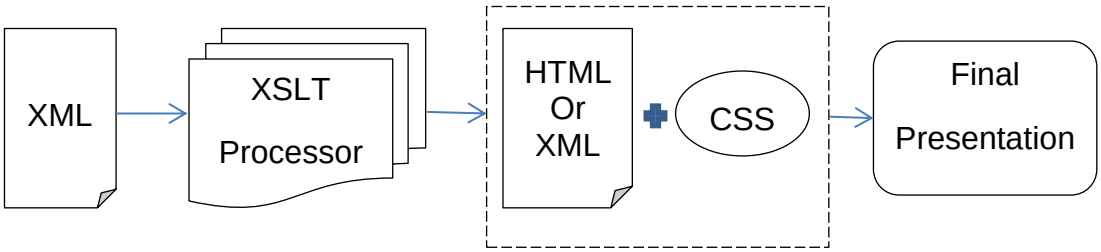

Figure 2-3. A simple example of an XSLT workflow.

2.4.2.4 JSON

JavaScript Object Notation (JSON) is a text-based format for representing simple data structures and objects and can be parsed by many programming languages. The primary use of the JSON format, which is based on a subset of JavaScript called ECMAScript, is for transmitting structured data over a

network connection. JSON is an important alternative to XML as it benefits from simplicity, especially when working with JavaScript (Zakas et al. 2007). Listing 2-5 shows a simple example of the JSON format to represent a class called “School.”

```
{ "School" :
   { "students" :
     [
      { "name" : "Adam Smith" ,
                 "age" : "28" }
     ]
   }
}
```

Listing 2-5. A simple representation using the JSON format.

2.4.2.5 The XMLHttpRequest Object

XML Http Request (XHR) is considered the core of AJAX as it sends data to a web server in HTTP/HTTPS format and receives the server response as XML, JSON, HTML, or plain text. The primary feature of XHR invokes the server with a request and receives the response in the background without the need to reload the web page. This process is achieved by initialising an HTTP request to the server via a send() method and obtaining the returned information (if it exists) via an available response method, such as responseXML() (Gosselin 2010). Table 2-1 shows the common properties and methods of the XHR Object.

Property or Method	Description
readyState	Integer indicating the state of the request, either: 0 (uninitialized) 1 (loading) 2 (response headers received) 3 (some response body received) 4 (request complete)
onreadystatechange	Function to call whenever the readyState changes.

status	HTTP status code returned by the server (e.g., “200” or “404”).
statusText	Full status HTTP status line returned by the server (e.g., “OK, No Content”).
responseText	The full response from the server as a string.
responseXML	A document object representing the server’s response parsed as an XML document.
abort()	Cancels an asynchronous HTTP request.
getAllResponseHeaders()	Returns a string containing all the HTTP headers the server sent in its response. Each header is a name/value pair separated by a colon, and header lines are separated by a carriage return/linefeed pair.
getResponseHeader(*header Name*)	Returns a string corresponding to the value of the *headerName* header returned by the server (e.g., request.getResponseHeader("Set-cookie")).
open(method, url [,asynchronous [, user, password]])	Initialises the request in preparation for sending to the server. The method parameter is the HTTP method to use, for example, “GET” or “POST”. The value of the method is not case sensitive. The URL is the relative or absolute URL to which the request will be sent. The optional asynchronous parameter indicates if send() returns immediately or after the request is complete (default is true, meaning it returns immediately). The optional user and password arguments are to be used if the URL requires HTTP authentication. If none are specified and the URL requires authentication, then the user will be prompted to enter it.
setRequestHeader(name, value)	Adds the HTTP header given by the name (without the colon) and value parameters.
send(body)	Initiates the request to the server. The body parameter should contain the body of the request, i.e., a string containing fieldname=value &fieldname2= value2, for POSTs or a null value for a GET request.

Table 2-1. Common properties and methods of the XMLHttpRequest Object (Powell 2008).

2.4.3 How AJAX Works

JavaScript initiates communication with the server through the XMLHttpRequest object to send the user request, which can be triggered by a user’s action or a time-driven event. The server-side process handles the request and, once complete, the server sends the resulting data back to the browser in XML, JSON or plain text format. At this stage, the browser uses JavaScript to access the DOM and update the user page dynamically. The entire process from the client to the server and back is performed behind the

scene meaning that no reload of the web page is required (Wang 2011). Figure 2-4 outlines the AJAX mechanism within a client-server communication. Listing 2-6 shows a simple implementation of how AJAX works.

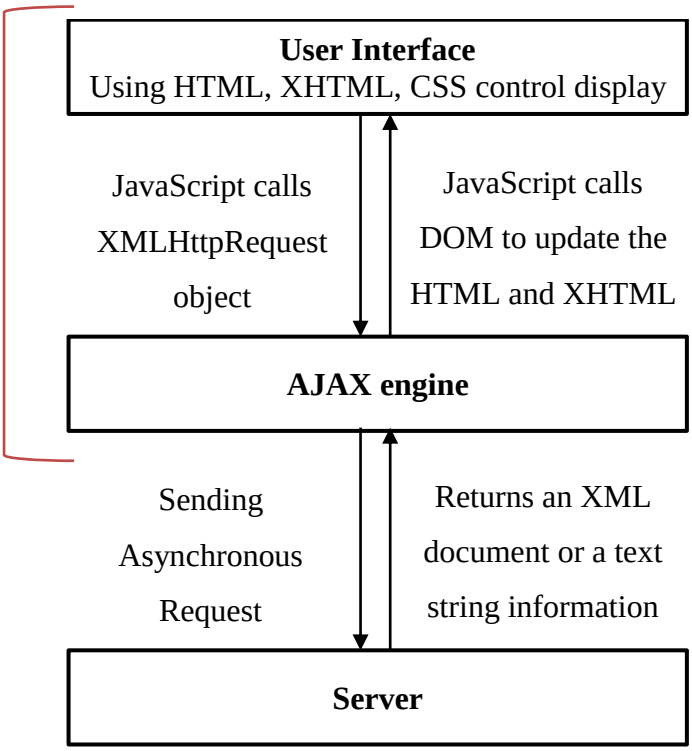

Figure 2-4. The AJAX mechanism.

```
<!DOCTYPE html>
<html>
<body>
<div id="demo">
<h2>The XMLHttpRequest Object</h2>
<button type="button" onclick="loadDoc()">Change Content</button>
</div>

<script>
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML =
      this.responseText;
    }
  };
  xhttp.open("GET", "show_details.php", true);
  xhttp.send();
}
</script>
</body>
</html
```

Listing 2-6. A simple implementation of AJAX.

2.4.4 The Advantages and Disadvantages of AJAX

Advantages: AJAX offers many features over standard web-based applications (McClure 2007), and the advantages are summarised in the following:

- AJAX can make asynchronous calls to a web server allowing the client to avoid waiting for all data to be received before the next action.
- Minimal data transfer: By not performing a full post-back and sending all form data to the server, network performance can improve.
- Limited processing on the server: Along with only necessary data being sent to the server, it is also not required to process all form elements.
- Responsiveness: Because AJAX applications are asynchronous on the client, they are perceived by users to be very responsive.

Disadvantages: As with all other web techniques, AJAX also includes the following drawbacks:

- AJAX is dependent on JavaScript, so if there is a JavaScript-related problem with the browser or OS, then AJAX functionality may not work.
- Search engines generally do not index the JavaScript present in web pages, so AJAX can be problematic for search engines as it incorporates JavaScript for much of its code.
- Pages dynamically created with AJAX do not adequately support the ‘back’ button on a browser, which is intended to return the user to the previous page.
- The data for all requests is URL-encoded, which increases the size of the request.

- There may be server overload with multiple requests, such as a request for each user keystroke.

2.5 Reverse AJAX and Real-time Web Technologies

Increasing demands on modern web applications require not only dynamic and responsive behaviours but also providing real-time information where web users are notified about events related to their interests as they occur. This high demand has pushed developers to find new solutions and tools that can work together to satisfy these features. A range of real-time web techniques have been introduced to the web environment over the last decade, and they vary from simple to sophisticated in terms of implementation. They are also adopted on a case-by-case basis depending on performance and efficiency needs. Common real-time web techniques, such as Polling, Comet, and WebSockets, are reviewed below to provide a foundation for their concepts, requirements, features, and drawbacks with a focus on potential improvements.

The target of Reverse AJAX is intended to make the server act with more responsiveness by pushing information to the client as it is available without the client explicitly making a request to the server. By default, AJAX requests can only be initiated from the client to the server. This limitation is resolved by using reverse AJAX techniques to provide responsive communication between the server and the client (Carbou 2017). The following techniques are representative of Reverse AJAX.

2.5.1 Polling

The concept of Polling is used in electronic communication where a controller device continually checks the linked devices' states as to if they are connected, want to communicate or use resources, such as a shared line (Rouse 2016). In the web environment of AJAX, the concept of Polling, based on the real-time Pull approach, handles client-server communication by requesting and sending data in real-time. In this technique, the browser sends requests to the server at regular and frequent intervals, for instance, every 3 seconds, to check if updates are available to be sent to the browser and presented in a designated part of the webpage. This process is implemented with the JavaScript function **setInterval()** (Carbou 2017), and Figure 2-5 demonstrates the Polling communication.

- **Advantages**: Polling is easy to implement and does not require special functionality on the server side, and it works within all browsers.
- **Disadvantage**: This method is rarely employed as it significantly increases the overhead of bandwidth and leads to lost resources.

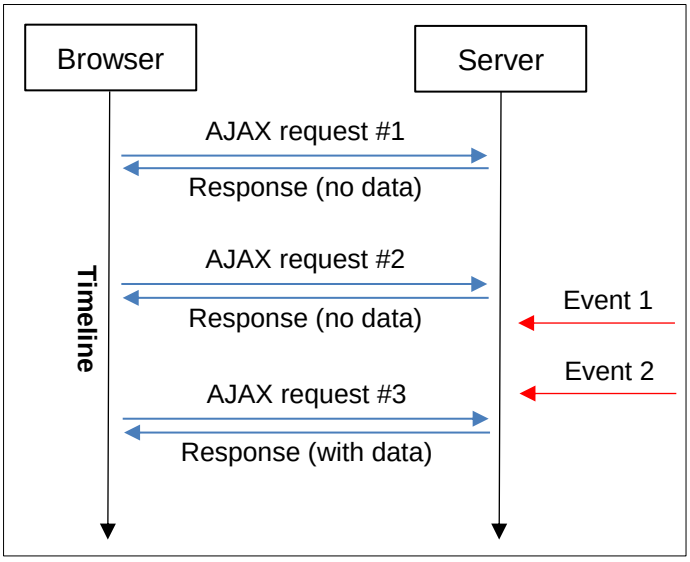

Figure 2-5. The Polling communication process.

2.5.2 Piggyback

In the Piggyback technique, the server, having an update to send, waits for the next time the browser makes a connection and then sends its update along with the response the browser is expecting (Carbou 2017). Figure 2-6 shows an example of the Piggyback technique communication.

- **Advantage**: With no requests to return data and since the client controls when it sends requests, fewer resources are consumed. This approach also works within all browsers and does not require special features on the server side.
- **Disadvantage**: It cannot be explicitly determined when events accumulated on the server side will be delivered to the client because it requires a client action to request.

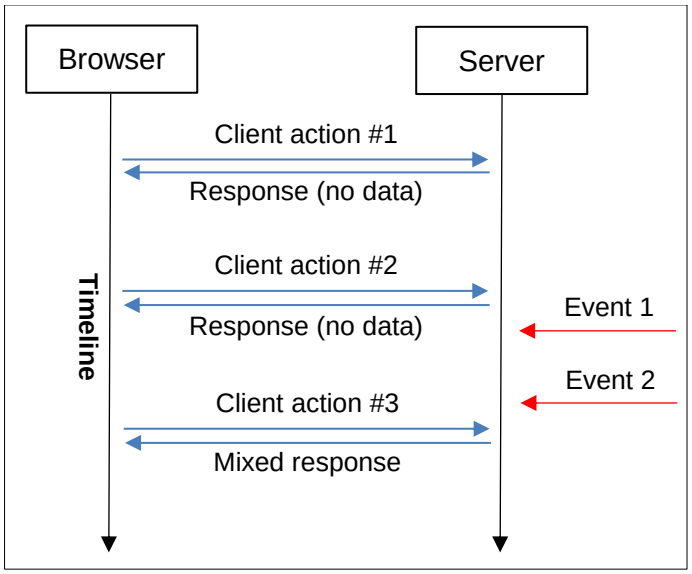

Figure 2-6. The Reverse AJAX process with Piggyback communication.

2.5.3 Comet

Comet (also known as long-lived HTTP or server push) lets the client send the AJAX request, and if no data is available to send back, then the server keeps

the connection alive until the request is completed and ready to be pushed back to the browser. In the case of a connection timeout, the client must make a new request to the server to accomplish the event (Campesato and Nilson 2011). Figure 2-7 shows an example of the Comet communication approach.

- **Advantage:** Comet reduces the load on the server by avoiding empty responses that can overload the bandwidth. In other words, it resolves the drawback of the Polling technique. Also, this long-living technique does not require browser plug-ins while providing real-time updates.

- **Disadvantages**: Regarding coding and implementation, Comet is not as simple as Polling and Piggyback, and also requires advanced features and configuration on the server (Dory et al. 2012).

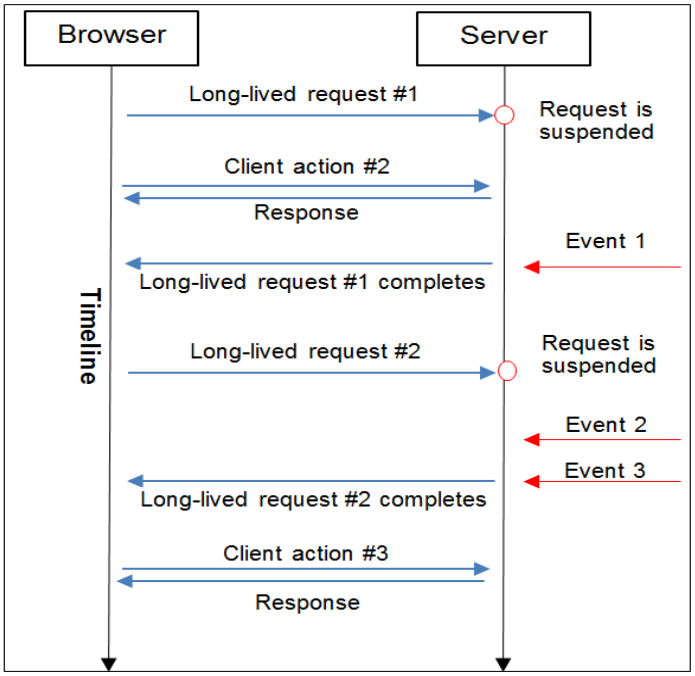

Figure 2-7. The Reverse AJAX process incorporating the Comet approach.

2.6 Forever Frames

The Forever Frames technique uses a hidden <iframe> element to communicate with the server with continuous requests. Once an event occurs, the server pushes back chunks of JavaScript tags that are executed in the browser in the same order as they were received (Gravelle 2017).

- **Advantages:** The Forever Frames technique is compatible with all standard browsers and offers a simple implementation.
- **Disadvantages:** When errors occur in this technique, they cannot be handled correctly as the code on the main page cannot control the connection. Another negative aspect is that some browsers indicate the page is still loading due to a continuous connection with the server. There may also be server load issues while maintaining the connections even if it is not delivering data (Johansen 2011).

2.7 Web Sockets

Web sockets is a dual-channel communication technology for web applications that was introduced in HTML 5. This concept does not rely on HTTP headers to communicate with the server so that the overhead is reduced to only a few bytes. A web socket starts by creating a WebSocket object and includes a URL starting with ‘ws’ or ‘wss’ (for the secure version) to make the connection (Kessin 2012).

- **Advantage:** Both the client and the server can initiate a data transfer at any time, which means the server can update without polling. In other words, data can be delivered without waiting for client requests ensuring a more responsive web application (Wessels et al. 2011).

- **Disadvantages:** Although Web Sockets are now supported on the most common browsers, there remains a compatibility issue with earlier browser versions. Also, the older standards for Web Sockets are disabled on some browsers for security concerns (Caniuse 2016).

2.8 Discussion

This chapter overviewed the most common tools and technologies needed for building real-time websites and web applications, including databases, client and server scripting languages, and techniques that support the concept of a real-time web (such as AJAX). The review considers several tools that can be used for understanding the mechanism of current real-time web approaches and for possible implementations of improvements. With the usability of MySQL for supporting cross platforms operating systems, including Linux, Microsoft, Macintosh, and UNIX, it offers an open source database management system that can be installed on a local machine with a server package for practical development needs. PHP integrates very well with MySQL and offers strong support from libraries as a server-side scripting language. JavaScript is nearly universally available on browsers and does not depend on special server technologies. AJAX is an essential foundation for several real-time techniques as it supports many rich Internet applications.

In addition to these reviewed technologies, this chapter investigates the different approaches of real-time client-server communication for building responsive real-time websites that can update users with events from the server as they occur. Their implementation, acuracy, compatibility, and efficiency are contrasted with various approaches as each is based on a

different concept, including the Pull concept, as in Polling, Push concept, as in Comet, and Dual-channel, as in WebSockets.

Each technique offers advantages and some drawbacks. Polling is simple to implement with low latency of updates. However, it can easily overload the server with unneeded connections. On the other hand, with Comet, while there is complicity when implementation and special settings required at the server, the approach is more efficient compared to Polling with very low latency. Piggyback, in contrast, is more efficient in terms of traffic overhead since no open communication is needed or redundant connections to the server, but a very high latency is likely. Forever Frames can be implemented with no issues across common browsers, but it offers challenges for tracing errors and unwanted browser indications may affect its reliability. WebSockets is a good choice for high traffic scenarios but must overcome compatibility issues with browsers along with some security issues because of the open connection maintained between the client and server.

A single, perfect technique for all web application cases does not currently exist, which suggests the possibility for improvements in these studied techniques for overcoming drawbacks while enhancing compatibility and efficiency of the use of resources.

2.9 Summary

Successful websites and web applications gain users' satisfaction with fast-loading, well-presented content, interactivity, and responsiveness. Moreover, the most successful and famous websites and web applications today, e.g. Facebook and Twitter, provide users with real-time data by keeping them

informed of the latest customised updates. This remarkable development of web applications is a result of various tools, languages, and techniques that integrate to accomplish building such valuable websites. This chapter focused on the most common related technologies and, specifically, the real-time techniques where a review suggests areas of possible improvement as each technique offers some issues that might be tackled.

Solutions may be explored in favour of reducing client-server communications by considering factors such as efficiency, compatibility, and simplicity of implementation. Two areas of development are highlighted. First, controlling inactive browsing, and second, enhancing the efficiency of client-server communication based on the Pull approach. For implementation purposes, tools such as MySQL, PHP, and JavaScript are considered to identify generic architectural models for web applications that feature multi-user change and update functionality, although solutions could be implemented in other ways.

The next chapter reviews solutions presented to improve current real-time web approaches as well as the key related work of these suggested areas of improvement. A detailed discussion about the differences between their concepts and mechanisms for enhancing current real-time web techniques will be provided to produce more efficient and reliable solutions with fewer drawbacks.

Chapter 3

3 Literature Review

3.1 Introduction

This chapter summarises related work and key papers that characterise different approaches for building real time web-based applications as well as suggested solutions to tackle issues, such as latency, header overload, and efficiency.

3.2 The Concept of Real-time Notifications on the Web

Basicevic et al. (2007) used a Publisher-Subscriber design pattern for a solution to realise communication services in a distributed Intrusion Detection Systems (IDS). This type of design pattern enables the central system to monitor the other nodes (i.e., systems) by notifying it of breaches of security. The Publisher-Subscriber design pattern, which is used for event notification, can be divided into three phases. The first is **subscription** where the association between the observer (subscriber) and observed entity (publisher) is created. The observer is then notified by the observed entity about its state changes. The second is **publication** when the publisher notifies the subscriber about its state changes. In the third phase of **termination,** the subscription between the observer and the observed entity is terminated. Figure 3-1 illustrates the interaction between the subscriber and publisher. This design pattern can build web-based applications that require real-time notifications or data updates.

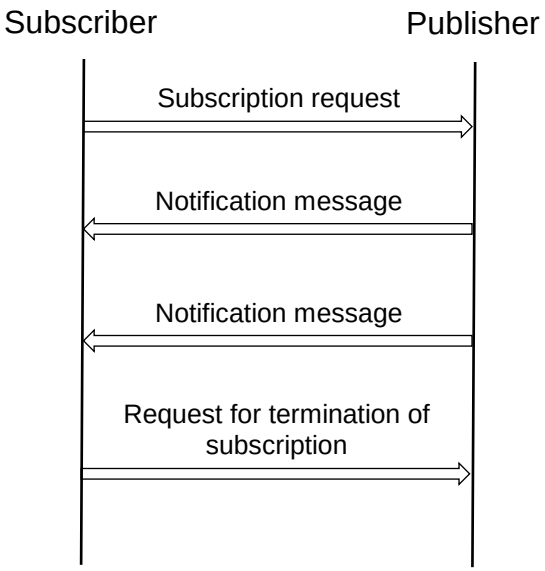

Figure 3-1. The interaction between the subscriber and publisher in the Publisher-Subscriber design pattern.

3.3 **AJAX push pattern**

Shen et al. (2009) introduced a GPS monitoring system based on AJAX real-time techniques with a focus on eliminating the drawbacks in existing systems that leverage AJAX. One problem is **redundancy** where the client uses a Polling model to access the server regularly during a non-optimal interval, which results in unnecessary client-server contacts. The second drawback is a **delay** that occurs with an increasing polling interval. To solve these two issues, they incorporated reverse AJAX or long-lived polling (e.g., Comet) to provide a nearly persistent connection with the server. Once the update becomes available, the server pushes it back to the client without waiting for its request. However, with this AJAX model, there still exists a drawback of **the lack of scalability** as the server holds an idle connection for each browser. Therefore, the ideal solution they provided uses Comet with Continuations supported by Jetty 6, which is a Java servlet that can process AJAX requests,

as well as DWR (Direct Web Remoting), which is a Java library that enables Java on the server to interact with JavaScript in the browser.

3.4 Push vs Pull Performance and Suitability as Real-time Web Techniques

Bozdag et al. (2009) compared the **Pull** and **Push** models designed for web-based real-time event notification and data delivery. In the Pull model, the client frequently requests the server for updates or notifications using a time function according to a fixed interval. The Push model utilised Comet and was implemented by two tools, COMETD and DWR. COMETD is a framework that supports the Comet protocol called BAYEUX, and DWR is a Java library that supports Comet. The authors' results show that the Push model achieves high data coherence and high network performance and can reliably serve a higher number of clients. However, when the number of users increases, some data misses will occur with Push.

On the other hand, in the Pull approach, data coherence cannot be achieved when the interval is longer than the publish interval. If it is less, then the network performance will be negatively affected. In some cases, Pull causes as much as seven times more network traffic compared to Push. Pull performs closely to Push only if the Pull interval equals the publish interval, which is challenging as the publish interval is rarely fixed. In other words, the Pull approach is useful only in situations where the data is published according to a regular pattern.

Bhide et al. (2002) presented work to address the coherency issue of disseminating data between server and clients according to client satisfaction

and tolerance of synchronised cashed data with server updates. They produced two techniques of PaP (Push and Pull) and PoP (Push or Pull).

The mechanism of PaP switches between Push and Pull adaptively to meet the client's cached data coherency satisfaction. It uses an adaptive TTR (Time To Refresh) technique to pull the server updates according to a calculated time with consideration to TCR (Temporal Coherence Requirement). The client sets this value, but in this technique, which utilises a registered proxy for each data item of the client's interest, can change to push instead if the updates are available at the server, so the subsequent pull will take time as it is aware of when the next refresh time.

For the PoP technique, it is assumed it can determine which method (Push or Pull) is adequate for clients to maintain the coherency of the cached data with server updates in line with the client requirement value. It is designed with the ability to classify clients to be provided with either the push or pull service, and this classification is dynamically changeable in response to the state of the current clients and network conditions. It initialises by registering clients with the Push protocol by default, and changes to the Pull service for some clients for the following scenarios:

- The state of the connection between the client and server is lost as a result of network errors.
- The occurrence of a resource constraint upon a new client registration or the change of network bandwidth.

Both techniques of PaP and PoP were supported by multiple Pull-based formulas that were introduced and implemented to enhance the client-server

Pull performance. These were then called the Adaptive TTR technique as it uses the Time To Refresh concept, which varies the refresh time for pulling data from the server in regards to the required temporal coherence by the client. The Adaptive TTR approach considers the following four factors:

1. The static value limits to avoid setting TTR values too high or too low.
2. The most rapid changes that have previously occurred.
3. The most recent changes to the polled data.
4. The estimate of the TTR value based on the most recent change to the data.

The above techniques focused on three points regarding the improvement of the real-time web. First, they address the client requirements for satisfying synchronous data. Second, they combine the Pull and Push protocols to achieve high performance with client satisfaction. Third, they improve the Pull-based real-time technique to be more responsive to the client's needs and server updates with enhanced usage of resources. This work relied on data-item values (i.e., market share values) to control the frequency of the server data updates based on the client's interests. Therefore, registered proxies and special server settings were necessary to obtain the expected results.

3.5 Controlling background operations and updates when a browser window or tab is inactive

According to Cimpanu (2017), Google Chrome (as of version 57) has limited background tab use to 1% of a CPU's core by delaying JavaScript timers for inactive tabs. Moreover, on mobile devices theses timers are entirely suspended after five minutes of being inactive with some exceptions for sites

using real-time communication, e.g., WebSockets. The user may still disable this property of the tab's control to allow background operations and updates to continue even if the focus is shifted to another tab. Chrome developers were encouraged to control timers in the background to reduce the consumption of resources and battery usage for devices such as smartphones, tablets, and laptops. Google is planning more development regarding background tab operations with the expectation that by 2020, Chrome will stop all JavaScript background operations after 'n' minutes without the ability for users to disable the feature.

The behaviour of web browsers varies in dealing with background tabs. Satheeq (2015) demonstrated that Firefox and Chrome limit the delay of the JavaScript timer (*setInterval*) to 1000 ms when the tab is out of focus, whereas the IE browser does not. The other function controlled on browsers is *Window.requestAnimationFrame*, which handles displaying animation continually by allowing an update every 'n' frames per second. This function is paused on Chrome and IE for inactive tabs while Firefox runs this function slower than the normal rate when tabs are inactive.

3.6 Discussion

According to the reviewed related work, we suggest that the concept of the Publisher-Subscriber pattern is beneficial when adopted for building a real-time notification system as it keeps the user notified with events in a real-time manner through the subscription to these events so that the publisher or server can send out the desired information.

In terms of real-time web-based applications, using Comet (the Push technique) is an ideal solution for monitoring systems and is a good choice compared to Polling, which requires high bandwidth. This solution remains limited to a targeted case, especially as it uses a collection of tools to suit that case, and requires specific settings and features to achieve sufficient client-server communication.

The concept of the Adaptive Push-Pull, according to Bozdag et al. (2009), appears promising theoretically for switching between the Push and Pull techniques, thus ensuring the use of the best of both as a single package solution. However, the approach must be implemented and tested before judging its reliability and scalability for a range of use cases. Moreover, the mechanism for changing from Pull to Push or vice versa may not be as simple as presented as the suggested algorithms are based on the server response rate.

Bhide et al. (2002) also presented valuable work regarding combining the Push and Pull techniques with improvements to the Pull concept. This work was focused on temporal client satisfaction of pushed real-time updates upon a data value limit control. Specifically, with the Pull technique, they used proxies to manage temporal coherence for each data value requirement in line with calculating the TTR range and value at the server, which involves additional time to calculate as well as the use of resources for each pull. This can result in overwhelming the server with extra processes for each client, especially when a large number of clients are served. Also, the use of the client's TCR, which is based on data values, can suit only some cases that depend on the limit of the data value itself. Therefore, these solutions must be limited to some

cases where the Pull technique is controlled by a range of values instead of change the time.

From the browser's perspective, real-time websites can be monitored and controlled at the browser side to help save resources and reduce unnecessary client-server connections. Google developed its well-known and widely-used browser to provide more control in inactive background tabs by delaying JavaScript's timers or stopping them completely after a certain period. This control can affect some real-time websites that need to operate even when their presentation is not active, which drove developers to abuse this type of control. On the other hand, this control is not applicable for websites that use WebSockets communication that require an open connection between client and server, which is unfair for other applications that use different real-time techniques. Moreover, the differences between web browsers in controlling the content and the operations of the background tabs have led developers to expect inconsistent performance and presentation of their websites. Related issues could affect user satisfaction, especially when dealing with animation, as explained by Satheeq (2015), where some browsers allow running animation in the background while others are completely stopped. For example with a specific user scenario, what happens if a user wants to listen to accompanying music to an animation that runs in the background? Therefore, this solution of controlling inactive tabs at the browser side is not always suitable, and while it may work with one browser, it remains an issue on another. So, another option for applying such control is necessary to ensure a consistent experience across all browsers without affecting the way websites are intentionally designed to function.

3.7 Summary

By considering all the techniques and solutions discussed in this chapter, one complete solution for all cases that requires real-time web communication may not exist. Selecting more than one technique may be adopted if it answers the demand or features desired in a website or web application, i.e., a website that requires real-time notifications, can be implemented using a technique of Push-based, Pull-based or both depending on several factors including latency, efficiency, and simplicity.

One solution is an improved Pull-based technique that incorporates the TTR concept. However, drawbacks in terms of simplicity and compatibility remain. Therefore, enhancing the Pull-based technique alternately while ensuring ease of implementation along high efficiency and expanding its use to all browsers and standard servers is intriguing. Also, for improving the efficiency and reducing unneeded use of resources, some solutions within the browser of controlling inactive tabs by applying timer delays and stopping other functions from running in the background appear reasonable approaches. Consequently, new issues arise such as inconsistent performance across browsers. The next chapter will focus on addressing these issues for developing a unique solution that offers fewer drawbacks.

Chapter 4

4 Browser Monitoring for Improving Real-time Web Communication

---

4.1 Introduction

Characteristics such as speed, responsiveness, efficiency, and real-time communication are desired today for good web applications. The most important feature for many web applications is how they can perform in real-time by serving users with their preferred or selected information and activities. To provide this service, different techniques may be adopted depending on the application needs in terms of currency, heavy loading, efficiency, and complexity.

This chapter focuses on enhancing the efficiency of these techniques by considering browser activity and if the user is interacting with a web page with real-time communication or working within another page. The proposed concept monitors browser activity to control the communication with the server in favour of reducing redundant client-server connections and enhancing the efficiency and use of resources. In other words, this approach minimises unnecessary communications and lessens the load on and use of server bandwidth.

It may be considered that bandwidth is less of a concern as its availability continues to increase. However, the needs for more bandwidth also appears

to be increasing. A recent estimate on the average size of a web page is over 2 MB (Finley 2016) with more than 62 images, scripts, and videos.

How much of a page is reloaded during a refresh depends on several factors, including the relative size of the elements, the use of AJAX to refresh portions, caching, and browser configuration. However, there is agreement that “what's consistent across almost every study of performance, though, is that pages are not getting significantly faster” (Pinto 2016). However, we also share that “the top ten most popular sites based on Alexa's rankings are significantly lighter than the average page, and they're getting lighter every month” (Finley 2016).

4.2 Controlling Inactive Browsing

Issues that arise in real-time web applications include the high volume of client-server connections to obtain updates from the server as well as redundant communication between the client and server, which may involve loading a large amount of data, while the user’s interest may be shifted and no longer following the current updates. These two issues can be applied to real-time techniques of Polling, Comet, and WebSockets. According to Satheeq (2015) and Cimpanu (2017), the browser side eliminates these issues by controlling the background operations by delaying timers and decreasing the speed of some JavaScript functions. The consequences of this approach led some developers to abuse these controls to ensure their websites work as intended. Websites can also have unexpected appearances on different browsers due to these solutions being applied differently across browsers as previously described in 3.6.

Our approach will consider handling inactive tabs from the developer side instead of the browser side. In other words, this solution will allow developers to control the background operations when the browser tab is not active to gain better use of resources and reduce server bandwidth while simultaneously avoiding problems that could occur if the browser applies these controls. Our concept monitors the user activity in the browser and determines if to continue showing updates and keep communications open between the browser and server or to postpone them while the user shifts their attention to another tab. Listing 4-1 shows a simple algorithm to observe user activity and control the connection to the server for receiving updates.

```
1  Setup the connection requirements
2  Establish connection
3  Send the first request to the server
4  Display results
5  Monitor user activity
   If (changed)
    If (browser/tab) still active then
       Continue contacting the server
    Else
       Postpone server connection

6  Go to step 5
```

Listing 4-1. An algorithm for observing user activity.

This model enhances the efficiency of client-server communication and can be implemented with any real-time web technique, including Polling, Comet, and Forever Frames, as it benefits from JavaScript features of monitoring browsing activity at the client side. Therefore, it reduces redundant connections to the server when the web page's browser tab is inactive.

To implement this solution, the Polling technique is selected as a test scenario to apply our concept as it offers the least complicated and most general real-time technique.

4.3 Enhanced Polling Prototype

The prototype's architecture is shown in Figure 4-1 and illustrates the mechanism of an enhanced Polling with a browser monitoring model. It functions by sending the first request to the server for any available updates, then beings to monitor the user activity at the browser. If the tab becomes inactive, then the client postpones sending requests to the server until the status of the tab changes to active when the communication between server and client resumes as before and updates are pushed back to the client. This prototype is implemented by controlling functions of client-side scripting written in JavaScript as it is widely used and supported by most browsers.

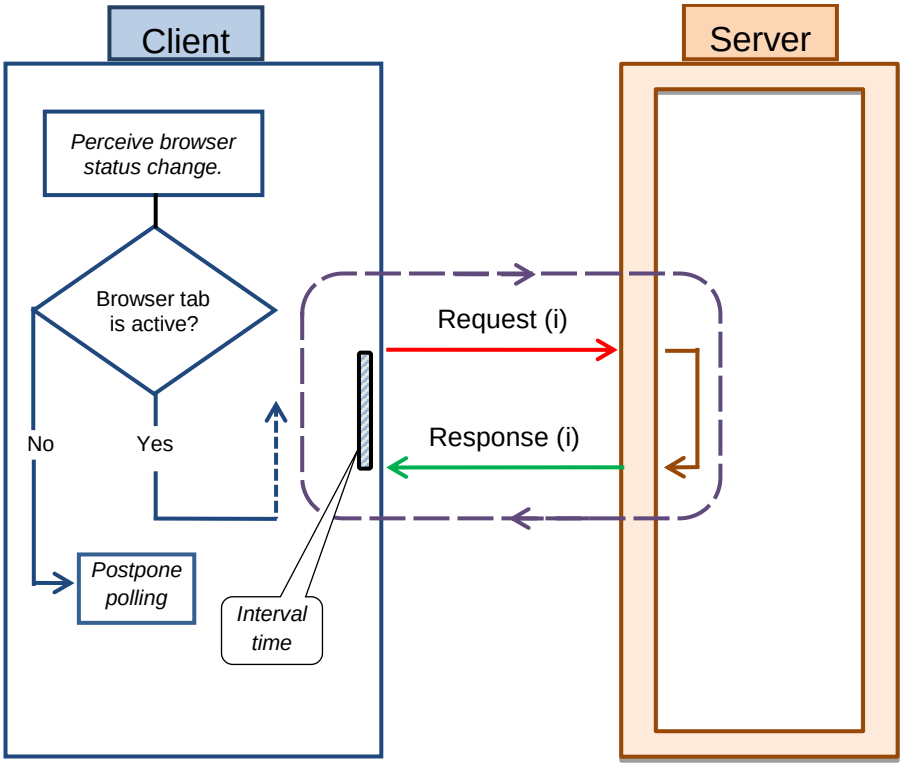

Figure 4-1. Prototype for active browser monitoring with Polling.

4.4 **Prototype Implementation**

A simple chat room web page was developed to represent our solution using HTML, CSS, PHP, and MySQL and implemented in JavaScript. The chat room application is based on the real-time Polling technique, which uses a JavaScript function to listen to the tap event. In the case of a 'focus' response (e.g., an active tap), then the function responsible for polling the server for possible updates (*continue_polling*) is called. On a 'blur' response (e.g., the tab is not active), then the calling the function (*postpone_polling*) responsible for stopping the send requests to the server is temporarily activated. Listing 4-2 shows the JavaScript functions that control communication with the server according to browser status.

```
<script>
var interval_id;
var intervals = 1000; // one second.

window.addEventListener("focus", function(event)
{ intervals = 1000; clearInterval(interval_id);
 interval_id = setInterval(function()
 { continue_polling();}, intervals); }, false);

window.addEventListener("blur", function(event)
{  postpone_polling();}, false);

function continue_polling()
{
 var xmlhttp;
 if (window.XMLHttpRequest)
 { // code for IE7+, Firefox, Chrome, Opera, Safari
   xmlhttp=new XMLHttpRequest(); }
 else
 { // code for IE6, IE5
   xmlhttp=new ActiveXObject("Microsoft.XMLHTTP"); }

 xmlhttp.onreadystatechange=function()
 { if (xmlhttp.readyState==4 && xmlhttp.status==200)
  {
   resultText = xmlhttp.responseText;
   document.getElementById("myDiv").innerHTML= resultText;
  }
}
```

```
document.getElementById("myDiv3").innerHTML=intervalTime;
xmlhttp.open("GET","myscript.php",true);
xmlhttp.send();
}

function postpone_polling()
 {
   clearInterval(interval_id);
 }
</script>
```

Listing 4-2. JavaScript functions for monitoring active browsing.

4.5 **Testing**

The proposed solution was tested by incorporating four users (User1, User2, User3, and User4) in a real-time conversation for 5 minutes. A Polling technique with an interval of every second was configured during which time all users were active except User4 who was intentionally not active for approximately three minutes. Figure 4-2 and Figure 4-3 present the web page views for User1 and User4 to compare the client-server communication and to illustrate how many connections were made to the server by each.

Your user ID User1

Enter your Comment now confused !!!

User ID	Comments	Time
User1	Hi everybody	12:00:11
User2	Hi there	12:00:20
User3	hi	12:00:45
User4	helloo	12:00:50
User2	Is there anyone using iphone?	12:01:02
User3	yes	12:01:12
User3	I've got iphone 6	12:01:27
User4	I'm not iphone's fan	12:01:47
User4	I've got samsung s7	12:02:10
User4	it's just perfect	12:02:24
User1	I heard it's nice and fast	12:02:46
User1	I'm thinking to buy s7 but currently have htc one	12:03:05
User3	iphone 6 is nice size and running owsum	12:03:20
User2	user1 you should think about iphone	12:03:32
User2	it's much better than samsung	12:03:50
User1	User4 ??	12:03:59
User1	are you still in	12:04:04
User3	yes I agree	12:04:15
User3	go for iphone	12:04:20
User1	I'll see, thanks for the advice!	12:04:32
User2	you'll not regret it	12:04:39
User1	thanks	12:04:46
User4	sorry been not in!	12:04:50
User4	don't listen to them!	12:04:55
User4	Samsung is the best ^ ^	12:05:01
User4	no jailbreak no hassle	12:05:11
User2	commooon	12:05:15
User3	no way	12:05:21
User1	now confused !!!	12:05:27
User4	hahaha	12:05:35

Number of connections == 325 times

Figure 4-2. The User1 screenshot of the browser monitoring model.

Your user ID User4

Enter your Comment hahaha

User ID	Comments	Time
User1	Hi everybody	12:00:11
User2	Hi there	12:00:20
User3	hi	12:00:45
User4	helloo	12:00:50
User2	Is there anyone using iphone?	12:01:02
User3	yes	12:01:12
User3	I've got iphone 6	12:01:27
User4	I'm not iphone's fan	12:01:47
User4	I've got samsung s7	12:02:10
User4	it's just perfect	12:02:24
User1	I heard it's nice and fast	12:02:46
User1	I'm thinking to buy s7 but currently have htc one	12:03:05
User3	iphone 6 is nice size and running owsum	12:03:20
User2	user1 you should think about iphone	12:03:32
User2	it's much better than samsung	12:03:50
User1	User4 ??	12:03:59
User1	are you still in	12:04:04
User3	yes I agree	12:04:15
User3	go for iphone	12:04:20
User1	I'll see, thanks for the advice!	12:04:32
User2	you'll not regret it	12:04:39
User1	thanks	12:04:46
User4	sorry been not in!	12:04:50
User4	don't listen to them!	12:04:55
User4	Samsung is the best ^ ^	12:05:01
User4	no jailbreak no hassle	12:05:11
User2	commooon	12:05:15
User3	no way	12:05:21
User1	now confused !!!	12:05:27
User4	hahaha	12:05:35

Number of connections == 148 times

Figure 4-3. The User4 screenshot of the browser monitoring model.

Figure 4-4 shows nearly 180 client-server connections were saved when User4 was not active on the browser in comparison with User1 who was active throughout the full duration of the test. Therefore, by applying our proposed solution to this real-time method, we improved the efficiency of the client-server communication in terms of redundant connections.

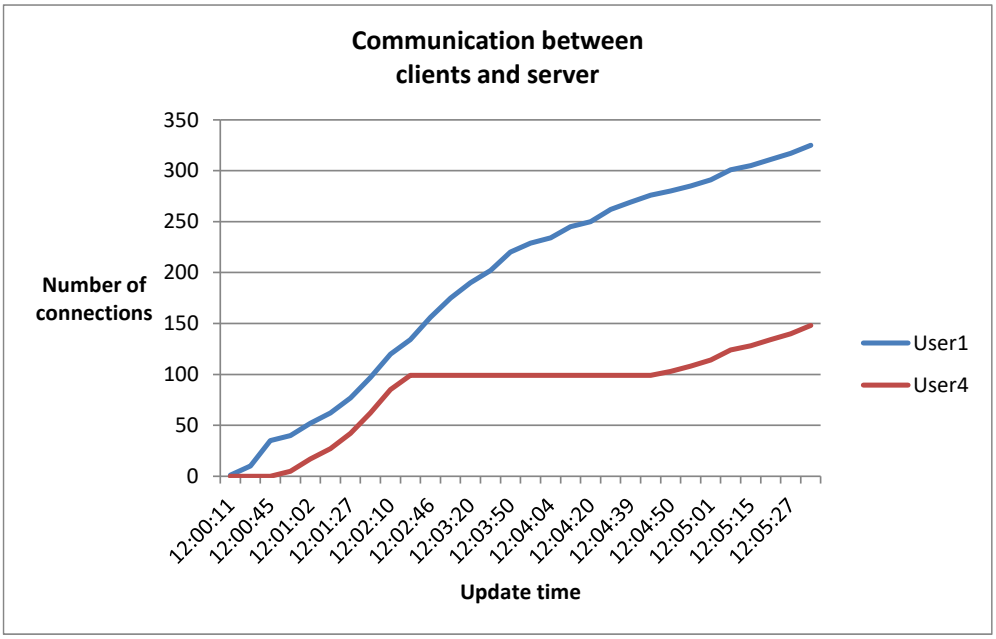

Figure 4-4. Client-server communications compared between User1 and User4.

4.6 Discussion

According to these test results, over half as many unnecessary client-server communications were avoided when applying the proposed concept of monitoring browser status. This solution, while simple, is very effective as it reduces client-server connections to a minimum level when users shift their interest to a different website or application. In other words, there is no need to continuously push updates (about a particular web page) to the client in the background as they can instead be re-triggered at once if the user returns focus to that web page. This significantly reduces the server bandwidth and improves the use of resources, such as CPU processes and battery consumption (for portable devices).

There remains a problem of generality in the other key area of monitoring user activity, which is based on the browser tab status in this context. Detecting user interest or attention in more complex ways tends to require more operating system and desktop manager resources, so is less general. Across

Unix systems, for example, there are different settings for Ubuntu and XFCE. Even some forms of focus detection may not be appropriate for certain desktops (Stackoverflow 2012). For example, commercial eye tracking approaches, such as Imotions (2017) and Tobii (2017) are starting to obtain levels of accuracy that might be useful in our area. Hachman (2016) reports some success navigating between browser tabs as well as developments from areas in assistive technologies for disabled people, such as Lupu et al. (2013). Most solutions appear to use specialised hardware, but the possibility of using the commonly integrated cameras of smartphones or laptops remains an area for further development.

Our model compared with the concept of background browser control presented by Satheeq (2015) and Cimpanu (2017) focuses on the developer side of enhancing real-time web applications instead of the browser side that forces websites to stop performing in the background, as in Chrome. The proposed model enables developers to choose to apply browser monitoring if appropriate for their website's needs and guarantees the consistent performance on all browsers. In contrast, browser-side control is applicable to some browsers, e.g. Chrome, but not all. Moreover, using browser-side control on Chrome excludes controlling background operations for some real-time websites, e.g. those based on WebSockets, whereas our model can be optimised over other real-time techniques for all browsers used.

4.7 **Summary**

Our principle addresses the need for more efficient real-time web applications regardless of the technique adopted while remaining practical in its implementation. We selected the Polling technique to clearly illustrate our approach based on monitoring the browsing activity at the client side to determine if the connection to the server should remain or be postponed. A prototype was produced and implemented using PHP, MySQL, JavaScript, and HTML with Polling as the real-time AJAX technique.

The testing results suggest our model saved a significant number of unneeded client-server connections when the user was not actively browsing the webpage. In other words, we avoid extended open connections or a large number of frequent connections to the server while the user's interest is shifted to another webpage or application. This solution also contributes to reducing data usage and saving resources, such as battery consumption, for websites accessed over smart devices and laptops.

Chapter 5

5 Adaptive Polling as an Enhanced Real-Time Web Technique

5.1 Introduction

The Web environment has witnessed a huge development in the last decade, producing many technologies that aim to enhance the interactivity and efficiency of Web applications, these are based on real-time web techniques that can fulfil the need for fast response to specific updates as they occur. Moreover, the featured real-time web applications is designed in the manner of (RIA) Rich Internet Applications; which have the appearance and functionality of desktop applications. (Kay 2016) .

As AJAX is one technology widely used by developers for building real-time web applications, its variant, Reverse AJAX (Crane et al. 2008), has the ability to provide real-time communications between the client and server through techniques of Polling, Piggyback, and Comet, as previously described in 2.9. These all provide the functionality of a responsive real-time web, but issues are experienced when these techniques are implemented for real use cases. This has triggered finding better solutions that can tackle these issues.

5.2 The Need for Improving Real-time Web Techniques

With several problems arising for real-time web applications, enhancing these techniques is of great value. Several solutions exist in the literature that can be adopted to build real-time web applications based on enhanced current real-time techniques or by merging more than one technique, such as combining Push and Pull techniques under a modified set of rules and conditions. (see 3.1)

These previously presented solutions cannot be ideal for all cases as one might resolve a latency issue while adding more complexity in implementation, or one may require servers with specific functionalities that affect using improved techniques with prevailing standards of resources and simplicity of execution. Pull was used by Bhide et al. (2002) as described in 3.4 to enhance the efficient use of resources and reduce the bandwidth of client-server communication. The suggested technique requires more processes at the server-side, such as using registered proxies and extra calculations to control the frequent pulls resulting in potentially overwhelming the server. Also, it considers only the client requirements of frequent updates based on data values, which limits using the technique to only cases when a value limit is needed during pulling the server updates. Thus, there remains room to develop the Pull-based technique differently way by considering the simplicity of implementation and the general standards of server requirements.

The focus in this chapter improves the Pull-based technique of Polling with more consideration to simplicity and fewer data transfers from the perspective of efficiency. We are motivated to finding a better alternative solution to suit

different situations where there are multiuser updates, especially when updates occur at irregular intervals of time as exemplified by auctions or hotel booking websites, for example. Therefore, our proposed technique can be a good option from the other solutions described above or even the dual-channel real-time technology ‘WebSockets’ that keep the client-server communication open in order to pass date in both directions, thus it remains limited as can face challenges with firewalls and other proxies while the other techniques benefit from HTTP functionalities for processing client-server communications, unless high data transfer is expected (Basta 2017).

5.3 The principle of Adaptive Polling

Adaptive Polling controls the interval of time for contacting the server so that it is flexible and changeable on demand. Depending on how frequent the updates are, the time can be increased or decreased based on server response. As seen in Figure 5-1, Polling carries a significant drawback in that a large number of requests (at a fixed frequency) to the server still has the potential of returning no data during most periods of the connection time. On the other hand, increasing the Polling interval causes a delay in retrieving updates from the server as they occur. This scenario suggests producing an alternative to Polling by enhancing its performance and making it more efficient and intelligent in terms of reducing the number of redundant requests to the server. This improvement is achieved by controlling the interval time of the client-server connections based on the frequency of actual updates from the server.

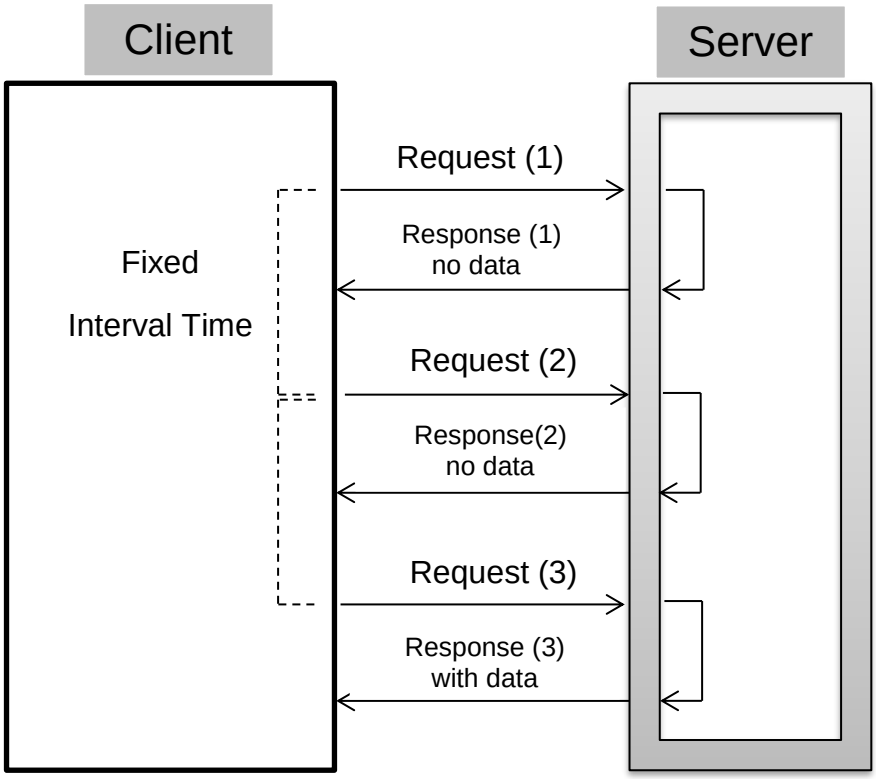

Figure 5-1. The Polling technique of AJAX.

5.4 **Adaptive Polling Prototype**

The mechanism of the Polling technique of AJAX is based on a fixed interval time to send requests to the server and receive replies if the update returns data or not. In Adaptive Polling, the interval time is not fixed but adjusts to the server response, which may include the consideration of time with a deadline as in an auction or plane ticket pricing. This means the interval time is reduced when approaching the time deadline. In this prototype, when the request returns no data from the server, the interval time is increased until it reaches a maximum, which can be set by the user's application. If the server responds with an update, then the interval time decreases until it reaches a minimum value, which can also be configured. With this dynamic approach, the efficiency of communicating with the server can be improved. Figure 5-2 illustrates this new mechanism of Adaptive Polling.

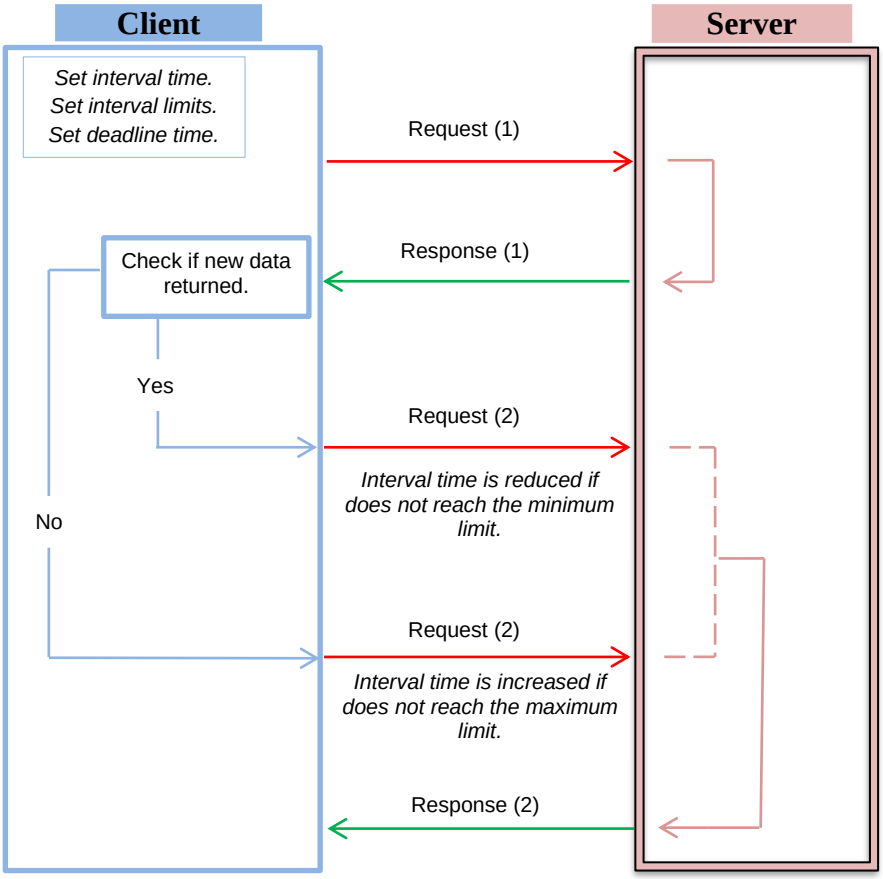

Figure 5-2. The Adaptive Polling prototype.

Listing 5-1 shows a simple algorithm to clarify the main processes of Adaptive Polling, which we implement with JavaScript and PHP.

```
Set the maximum interval  Max_int
Set the minimum interval  Min_int
Set the interval time  Interval
Set deadline time (if applicable!)
Repeat
{ RequestUpdates(Interval)
     Results = Call the function Check_for_update( )
     If Results = true [or current-time close to deadline-time] then
          Call the function ChangeInterval(true)
     Else
          Call the function ChangeInterval(false)
} while(true)

Function ChangeInterval(flag)
{
If flag = true then
   Decrease Interval while it is greater than Min
Else
   Increase Interval while it is less than Max
}
```

Listing 5-1. The Adaptive Polling algorithm.

5.5 Adaptive Polling Scenarios

Two scenarios are presented for controlling the interval time for our Adaptive Polling implementation, as are described below.

Scenario 1

The interval time increases by one second when the server responds with no update, and, equally, decreases by one second when the server answers with an update. Figure 5-3 illustrates this interval time modification for Scenario 1.

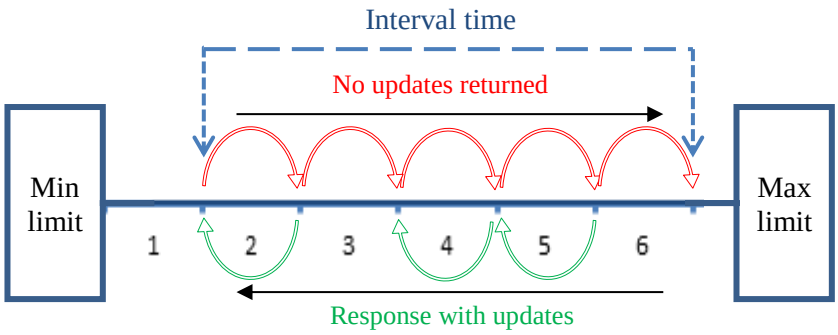

Figure 5-3. The Adaptive Polling Scenario 1 of controlling the interval time.

Scenario 2

The interval time increases by one second when the server responds with no update. If the server answers with an update, then the interval time decreases half-way to the minimum limit when the current interval is greater than one second as follows:

- If the current interval time is an odd number, then the interval adjusts according to i = (i+1)/2.
- If the current interval time is an even number, then the interval adjusts according to i = i/2.

Figure 5-4 illustrates the interval time modification process for Scenario 2.

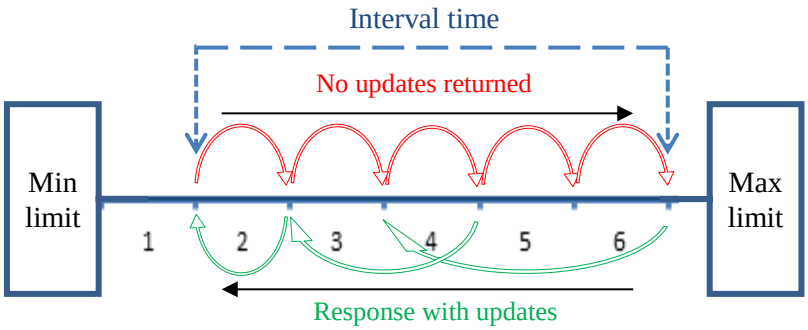

Figure 5-4. The Adaptive Polling Scenario 2 of controlling interval time.

The difference between these scenarios is that when the server answers with an update, the interval time in Scenario 1 decreases gradually for each update until reaching the minimum while in Scenario 2, it decreases half-way to the minimum from the current interval.

5.6 Implementing Adaptive Polling

Our implementation is based on the prototype in Figure 5-2 and interval control model of Scenario 1 presented in the previous section. We built a simple multi-user web interface to represent the mechanism of Adaptive Polling. Figure 5-5 presents the simple chat web page utilised where any user can add comments to the chat room. Also, it shows the current interval time in milliseconds, and the number of times it has contacted the server for potential updates. These requests are made at different frequencies depending on the user action of adding comments to the chat room. So, when a chat user initiates a change to the database through an interaction, the Polling interval time is decreased. When the server responds with no new data, the interval time increases until it reaches the maximum limit configured for the interval time.

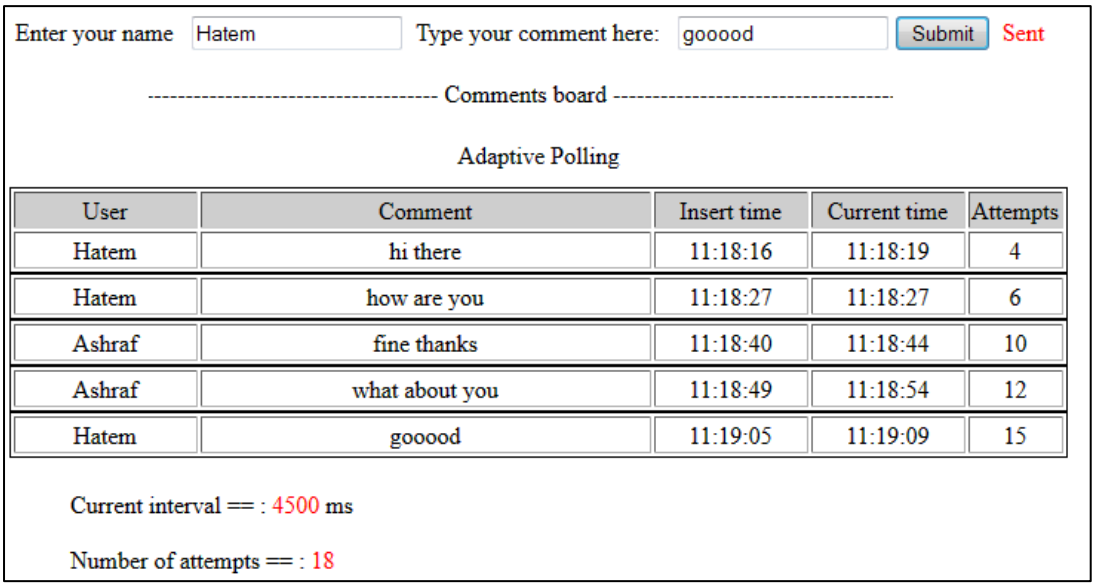

Figure 5-5. The Adaptive Polling test implementation.

Our goal is to identify generic architectural models for web applications that feature multiuser changes and update. We used JavaScript and XML as they are well supported for test implementations since they require no special browser or server features, although solutions could be implemented in other ways. Listing 5-2 shows the main JavaScript function responsible for calling the PHP script on the server to check for new updates, and Listing 5-3 shows the interval change function that increases or decrease the interval time in response to the server update.

```
function loadXMLDoc()
  {
    var xmlhttp;
    if (window.XMLHttpRequest)
       xmlhttp=new XMLHttpRequest();
    else
    xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");

    xmlhttp.onreadystatechange=function()
    {
       if (xmlhttp.readyState==4 && xmlhttp.status==200)
       { counter1 += 1;
         document.getElementById("myDiv2").innerHTML= counter1;
         if(xmlhttp.responseText != "")
            if(counter1 < 2)
              resultText = xmlhttp.responseText;
            else
```

```
            {
              resultText += xmlhttp.responseText;
              document.getElementById("myDiv").innerHTML=
              resultText;
              changeInterval(false);
            }
          else
           changeInterval(true);
        }
    }
       document.getElementById("myDiv3").innerHTML=intervalTime;
       xmlhttp.open("GET","myscript.php?ct="+counter1 ,true);
       xmlhttp.send();
   }
```

Listing 5-2. The check-for-updates function.

```
function changeInterval(flag)
 {
    if (!flag)
    {
      if (intervalTime > minInterval)
      {
        intervalTime -= intervalValue;
        clearInterval(si);
        si = setInterval(function(){ loadXMLDoc();},intervalTime);
      }
    }
    else
    {
      if (intervalTime < maxInterval)
      {
        intervalTime += intervalValue;
        clearInterval(si);
        si = setInterval(function(){ loadXMLDoc();},intervalTime);
      }
    }
 }
```

Listing 5-3. The change-interval function.

5.7 **Adaptive Polling vs Polling**

A simple simulation web page is implemented to compare the performance of the two techniques for illustrating the improvement offered by Adaptive Polling over Polling. This simulation allows multiple users to participate in a chat room and see their comments through both processes of Adaptive Polling and

normal Polling. This side-by-side approach allows for a visualisation of the difference between the two regarding the frequency of server connections and the update delay. Research students performed a simple test with the results presented in Figure 5-6 and Figure 5-7.

Adaptive Polling

User	Comment	Insert time	Current time	Attempts
Hatem	Hi there	17:08:36	17:08:40	3
Hatem	Hellooo	17:08:48	17:08:50	6
Abdo	Hi	17:09:02	17:09:02	9
Abdo	how are you	17:09:08	17:09:11	11
Hatem	goood	17:09:24	17:09:25	14
Hatem	where is the others	17:09:33	17:09:34	16
Ashraf	here man	17:09:47	17:09:50	19
Ashraf	how is it going	17:10:00	17:10:06	22
Hatem	good	17:10:16	17:10:18	24
Abdo	gooood	17:10:28	17:10:30	26
Hatem	where is Ibrahim and Muktar	17:10:52	17:10:55	30
Ibrahim	Yes I m here man	17:11:16	17:11:20	34
Ibrahim	how are you all	17:11:27	17:11:33	36
Hatem	oh nice to see u here Ibrahim	17:11:47	17:11:48	38
Ashraf	How r u Ibrahim where are you	17:12:00	17:12:01	40
Ibrahim	buzy a little bit	17:12:15	17:12:15	42
Ibrahim	what about you Ashraf	17:12:29	17:12:35	45
Ashraf	I m fine coping	17:13:06	17:13:12	51
Muktar	sorry guys was stuck	17:13:43	17:13:44	56
Muktar	Hatem is that ok	17:14:01	17:14:04	59
Hatem	yes that s ok Muk. Don t worry	17:14:17	17:14:17	61
Hatem	Thank you all guys	17:14:33	17:14:37	64
Hatem	for responding	17:14:44	17:14:50	66
Hatem	and see you later	17:14:57	17:15:03	68
Muktar	No prob man	17:15:16	17:15:18	70
Muktar	that s fine	17:15:30	17:15:31	72
Abdo	it s alright man	17:15:56	17:15:57	76
Ashraf	see ya	17:16:21	17:16:23	80
Ibrahim	see you later	17:16:34	17:16:36	82

Current interval: == **2000** ms
Number of attempts: == **82**

Figure 5-6. The Adaptive Polling simulation.

Polling

User	Comment	Insert time	Current time	Attempts
Hatem	Hi there	17:08:36	17:08:38	5
Hatem	Hellooo	17:08:48	17:08:48	15
Abdo	Hi	17:09:02	17:09:02	29
Abdo	how are you	17:09:08	17:09:09	36
Hatem	goood	17:09:24	17:09:25	52
Hatem	where is the others	17:09:33	17:09:33	60
Ashraf	here man	17:09:47	17:09:48	75
Ashraf	how is it going	17:10:00	17:10:01	88
Hatem	good	17:10:16	17:10:16	103
Abdo	gooood	17:10:28	17:10:29	116
Hatem	where is Ibrahim and Muktar	17:10:52	17:10:53	140
Ibrahim	Yes I m here man	17:11:16	17:11:16	163
Ibrahim	how are you all	17:11:27	17:11:28	175
Hatem	oh nice to see u here Ibrahim	17:11:47	17:11:47	194
Ashraf	How r u Ibrahim where are you	17:12:00	17:12:01	208
Ibrahim	buzy a little bit	17:12:15	17:12:15	222
Ibrahim	what about you Ashraf	17:12:29	17:12:29	236
Ashraf	I m fine coping	17:13:06	17:13:07	274
Muktar	sorry guys was stuck	17:13:43	17:13:43	310
Muktar	Hatem is that ok	17:14:01	17:14:01	328
Hatem	yes that s ok Muk. Don t worry	17:14:17	17:14:18	345
Hatem	Thank you all guys	17:14:33	17:14:33	360
Hatem	for responding	17:14:44	17:14:44	371
Hatem	and see you later	17:14:57	17:14:58	385
Muktar	No prob man	17:15:16	17:15:17	404
Muktar	that s fine	17:15:30	17:15:30	417
Abdo	it s alright man	17:15:56	17:15:57	444
Ashraf	see ya	17:16:21	17:16:21	468
Ibrahim	see you later	17:16:34	17:16:34	481

Current interval: == **1000** ms

Number of attempts: == **481**

Figure 5-7. The Polling simulation.

According to the results of the test, we find that Adaptive Polling only required 1/6 of the contacts to the server compared to Polling as Adaptive Polling sent 82 requests for updates and Polling sent 481 contacts. Therefore, there was a significant amount of redundant connections made by Polling compared to Adaptive Polling, which performed more efficiently from this perspective.

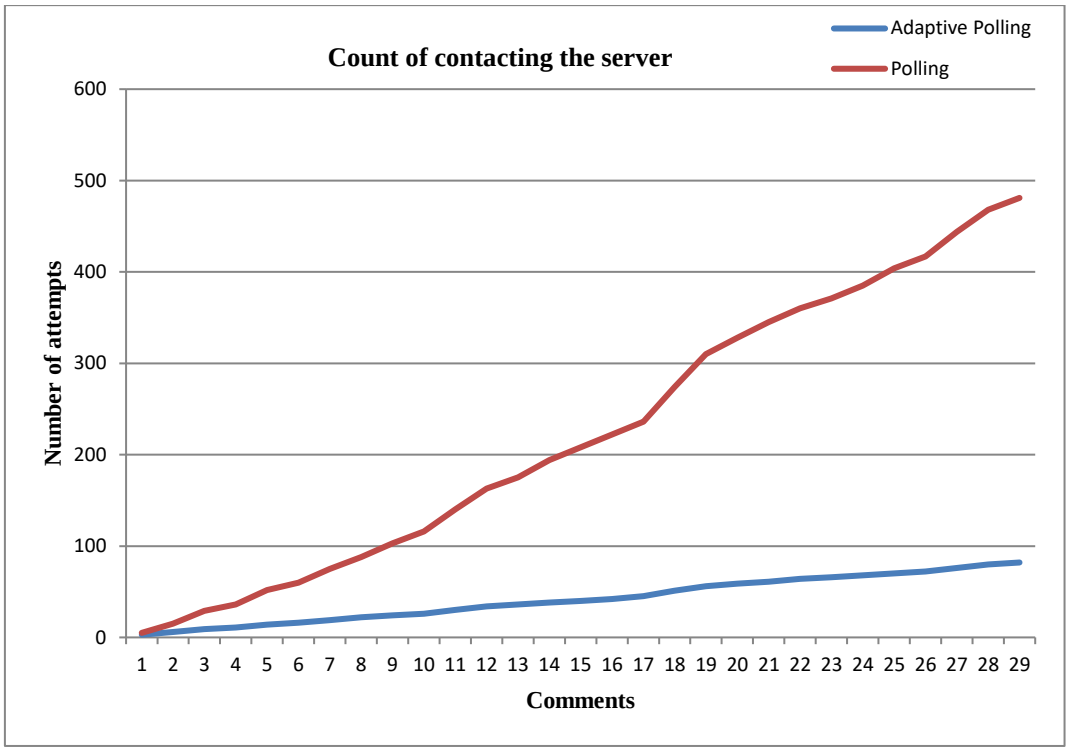

Figure 5-8. The client-to-server communication count according updates.

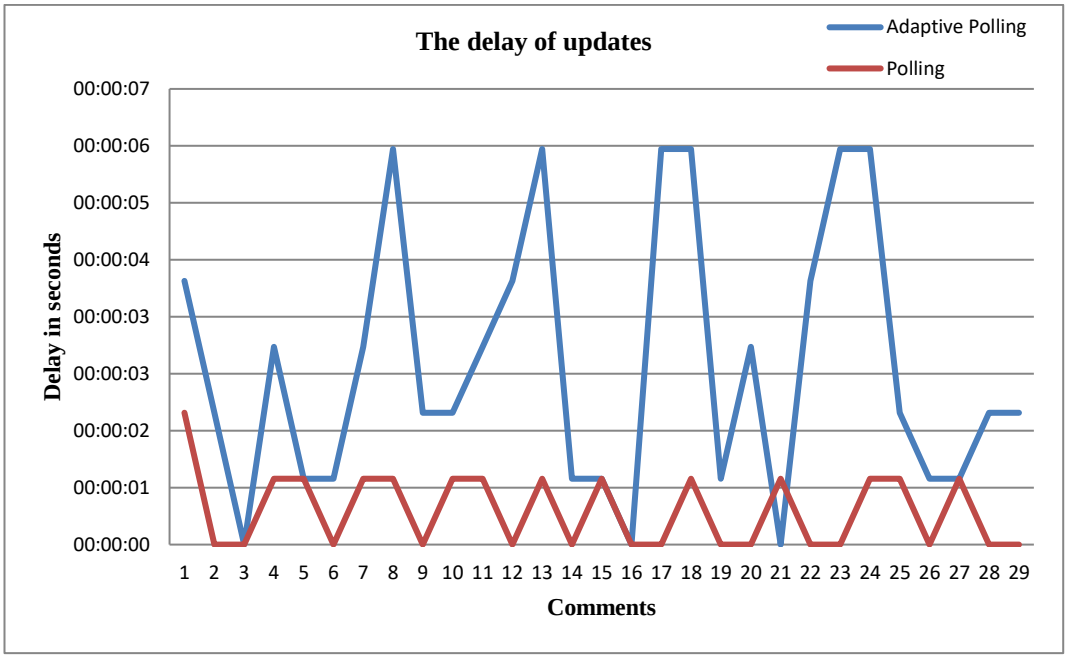

Figure 5-9. Time delays for retrieving updates from the server.

Figure 5-8 illustrates the number of communications required from both techniques. In terms of latency, Adaptive Polling includes more delay than normal Polling as shown in Figure 5-9. However, for many cycles they share

the same response time for an update while zero delay occurs when updates become available the same time as the server receives the request. This characteristic is a positive feature when considering the fewer number of times the client connects to the server.

5.8 Discussion

The performed test comparing Adaptive Polling and Polling indicates that the former avoids unneeded requests that do not return new data, which is especially valuable when no updates are expected for an extended period. While these results are only for a short time for the testing period, extending the proposed concept to a larger scale is expected to present a significant impact on reducing bandwidth requirements and enhancing the use of server resources.

On the other hand, Adaptive Polling does institute some latency not observed in Polling when after a decrease in the update frequency is set for some time and then suddenly accelerates. This minor latency does not affect the overall performance of the technique, and may not be observable in practical use cases. In addition, Adaptive Polling offers flexibility that enables the user to control the interval time for requesting updates by selecting from preassigned values when browsing the web page.

According to results of implementing and testing Scenario 1 of Adaptive Polling, Scenario 2 is expected to reduce the possible latency that can be experienced of applying Scenario 1 in some cases where periodically irregular updates; by shortening the interval time quickly when the frequency of updates accelerates suddenly after being quite for a period. This therefore can be more

suitable in such cases where the range of interval time of contacting the server can be widened without causing delay once updates occur faster at some time; presumably, if the minimum interval is 1 second and the maximum is 20 seconds and the current interval is the highest, which is 20 seconds, as no updates received for a period, then the server started to respond with new data regularly (let us say every 3 seconds), here with Scenario 2 the interval time will be reduced noticeably from 20 to 10 to 5 to 3 seconds instead of gradually, in order to reduce the delay of showing updates.

Although there is in general some similarity between Adaptive Polling and the Pull-based technique proposed by Bhide et al. (2002) and described in 3.4 and 3.6, the mechanism and concept of the technique are very different as listed in Table 5-1.

Description	Adaptive Polling	Pull-based technique by Bhide et al. (2002)
Needs change for server push capability or HTTP protocol	No	Yes
Uses registered proxies to work with the server	No	Yes
Calculation of the current interval	Processed at the client side.	Processed at the server side.
Time of calculation	Very short time.	Additional time involved when working with proxies and data values.
Implementation	Easy to implement.	Implementation can be complicated.
Compatibility	Works on all browsers and servers with no extra settings.	Not compatible with all servers, and extra settings and features are required.
The user can control the interval time	Optional, by applying interval time limits.	Required, by applying data value limits over proxies.

Table 5-1. Properties comparison of Adaptive Polling and another Pull-based technique.

5.9 Summary

The AJAX technique of Polling was improved by eliminating the high bandwidth and loss of resources by dynamically adjusting the interval time used for sending requests for updates from the server. This modification is performed at the client side but is based on if the server response returns an update. This enhanced technique, called Adaptive Polling, is implemented and tested directly against traditional Polling with results demonstrating that redundant connections were significantly reduced while expressing reasonable latency. So, this solution is a good alternative that can be implemented easily to work with all browsers and servers offering a suitable technique for many use cases, especially when updates occur irregularly.

Chapter 6

6 Conclusion and Future work

6.1 Conclusion

This thesis investigated real-time web technologies for their vital role in developing websites and web applications that serve users with data and information as they occur in real-time. The research studied the most common real-time web techniques along with suggested solutions that attempted to address drawbacks and issues of these techniques, such as latency, server bandwidth, and efficiency, suggesting areas for improvements.

The primary challenge was to improve the efficiency of these techniques by reducing client-server communication in conjunction with simplicity and compatibility within the framework of a standard web environment. Two models were introduced of browser monitoring and Adaptive Polling. First, the browser monitoring model monitors user activity at the browser to determine if it should to continue client-server communication or postpone it due to inactive browsing. This model was implemented and tested with the real-time web technique of Polling. Results suggested this solution significantly decreased the number of client-server connections by controlling background browsing, thereby reducing the bandwidth and improving the use of resources such as fewer CPU processes, less data usage, and enhancing battery consumption. Moreover, this model is designed as a developer side solution, which is an alternative to those that target only some browsers configurations, such as in

Chrome. Therefore, the model works for all browsers and can be optimised over different real-time techniques including Comet and WebSockets, ensuring consistency and compatibility within the web environment of browsers and servers.

Second, the Adaptive Polling approach is a Pull-based real-time web technique for enhancing the efficiency of Polling by addressing its drawbacks. This model reduces the number of unneeded client-server connections when no updates are returned from the server. This is accomplished at the client side by controlling the interval time of sending requests to the server based on its previous response. In other words, this decelerates the frequency of client-server connections when no updates are pushed back to the client. This solution was implemented as a web page with a real-time, multi-user chat room tested in direct contrast with traditional Polling to demonstrate its advantages. The results demonstrated the system avoided a significant amount of unnecessary client-server connections, which indicates that Adaptive Polling can replace the basic Polling technique, especially when no regular updates occur from the server. Furthermore, despite similarities with another Pull-based technique, Adaptive Polling is distinguished by its simplicity of implementation and compatibility with all browsers and servers with no special requirements or adjustments. Therefore, it is an excellent alternative to other real-time web techniques, such as Comet and WebSockets, in terms of considering the value of less bandwidth over minimal latency.

6.2 Future Work

This research introduced two models as real-time solutions for enhancing existing techniques and proved the validity of these concepts that can be easily optimised for building efficient, real-time websites and web applications. The next stage is testing these techniques in real, large-scale implementations for case studies across domains by allocating users randomly to different implementations. Additional future directions of this research may include the following:

- To improve the user activity detection for the browser monitoring model, consider using cameras and sensors in laptops and smartphones to offer additional monitoring touch-points for identifying inactive browsing.
- To optimise the browser monitoring model over other real-time techniques, appreciate its effectiveness in conjunction with its applicability and consistency despite the browser.
- Implement the Adaptive Polling approach according to Scenario 2 in cases of irregular updates over periods of time is expected, and test it within a large-scale environment.
- Further develop Adaptive Polling by combining it with the browser monitoring model as one solution and test it in a case study.

7 References

Allen, R. (2009). *Web development with JavaScript and Ajax illuminated*, Sudbury, Mass.: Jones and Bartlett.

Aziz, H. and Ridley, M. (2016). Adaptive Polling for Responsive Web Applications. In *Information Science and Applications (ICISA) 2016*. Springer, pp. 1157–1167.

Aziz, H. and Ridley, M. (2017). Real-time web applications driven by active browsing. In *Internet Technologies and Applications (ITA), 2017*. IEEE, pp. 4–8.

Basicevic, I.Popovic, M.and Kovacevic, V. (2007). Use of publisher-subscriber design pattern in infrastructure of distributed ids systems. In *Networking and Services, 2007. ICNS. Third International Conference on*. IEEE, p. 56.

Basta, M. (2017). What should you not use WebSockets for. *Quora*. Available at: https://www.quora.com/What-should-you-not-use-WebSockets-for [Accessed February 27, 2018].

Beal, V. (2013). CGI. Available at: http://www.webopedia.com/TERM/C/CGI.html [Accessed January 5, 2017].

Beighley, L. (2010). *jQuery for dummies* For dummie., Hoboken, NJ: Wiley Pub., Inc.

Bhide, M. et al. (2002). Adaptive push-pull: disseminating dynamic Web data. *IEEE Transactions on Computers*, 51(6), 652–668.

Bozdag, E.Meshah, A.and van Deursen, A. (2009). Performance Testing of Data Delivery Techniques for Ajax Applications. *Journal of Web Engineering*, 8(4), 287–315.

Brunner, R.J. (2003). *JSP : practical guide for Java programmers*, Amsterdam ;

London: Morgan Kaufmann.

Campesato, O. and Nilson, K. (2011). *Web 2.0 fundamentals with Ajax, development tools, and mobile platforms*, Sudbury, Mass.: Jones and Bartlett Publishers.

Caniuse (2016). Web Sockets. Available at: http://caniuse.com/websockets [Accessed October 12, 2016].

Carbou, M. (2017). Reverse Ajax. Available at: http://www.ibm.com/developerworks/library/wa-reverseajax1/ [Accessed February 11, 2017].

Chromatic (2014). *Modern Perl* Fourth edi., Onyx Neon Press.

Cimpanu, C. (2017). Chrome 57 Limits Background Tabs Usage to 1% per CPU Core. *bleepingcomputer*. Available at: https://www.bleepingcomputer.com/news/software/chrome-57-limits-background-tabs-usage-to-1-percent-per-cpu-core/ [Accessed November 25, 2017].

Crane, D.McCarthy, P.and Tiwari, S. (2008). *Comet and Reverse Ajax the next-generation Ajax 2.0*, Berkeley, CA. New York, N.Y.: Apress .

Dory, M.Parrish, A.and Berg, B. (2012). *Introduction to Tornado*, Beijing ; Sebastopol, Calif.: O'Reilly.

Duckett, J. (2008). *Beginning web programming with HTML, XHTML, and CSS* 2nd ed., Indianapolis, Ind: Wiley.

Dykes, L. and Tittel, E. (2005). *XML for dummies* 4th ed., Hoboken, N.J.: Wiley.

Evjen, B. (2006). *Professional ASP.NET 2.0* Special., Indianapolis, IN: Wiley Pub.

Finley, K. (2016). The Average Webpage Is Now the Size of the Original Doom. Available at: https://www.wired.com/2016/04/average-webpage-now-size-original-doom/ [Accessed December 5, 2016].

Gosselin, D. (2010). *JavaScript* 5th ed., Boston: Thomson / Course

Technology.

Gravelle, R. (2017). Comet Programming: the Hidden IFrame Technique. Available at: http://www.webreference.com/programming/javascript/rg30/index.html [Accessed February 10, 2017].

Gross, C. (2006). *Ajax patterns and best practices*, Berkeley, CA: Apress.

Gulipalli, G. (2015). 8 Reasons Why XML is a Versatile Data Conversion Language. *Invensis*. Available at: https://www.invensis.net/blog/data-processing/8-reasons-why-xml-is-a-versatile-data-conversion-language/ [Accessed December 5, 2017].

Gunelius, S. (2014). The Data Explosion in 2014 Minute by Minute – Infographic. Available at: https://aci.info/2014/07/12/the-data-explosion-in-2014-minute-by-minute-infographic/ [Accessed February 9, 2017].

Hachman, M. (2016). Tobii eyeX review: The “eye mouse” is magical, but just not for everyone. Available at: http://www.pcworld.com/article/3014523/peripherals/tobii-eyex-review-the-eye-mouse-is-magical-but-just-not-for-everyone.html [Accessed January 14, 2017].

Hanegan, K. (2001). *Custom CGI scripting with Perl*, New York ; Chichester: Wiley.

Hégaret, P. Le (2017). Document Object Model. Available at: http://www.w3.org/DOM/ [Accessed March 12, 2017].

Holzner, S. (2007). *Ajax bible*, Indianapolis, Ind.: Wiley.

Imotions (2017). Eye Tracking Hardware and Software Solutions. Available at: https://imotions.com/eye-tracking/ [Accessed January 21, 2017].

“Introduction to Ajax” (2016). Introduction to Ajax for Java Web Applications. *Netbeans*. Available at: https://netbeans.org/kb/docs/web/ajax-quickstart.html [Accessed December 10, 2017].

Johansen, C. (2011). *Test-driven JavaScript development*, Boston, Mass. ; London: Addison-Wesley.

Kahate, A. (2009). *XML & Related Technologies*, Delhi, India: Dorling Kendersley.

Kay, R. (2016). Rich Internet Applications. Available at: http://www.computerworld.com/article/2551058/networking/rich-internet-applications.html [Accessed November 25, 2016].

Keogh, J.E. (2005). *JavaScript demystified*, New York: McGraw-Hill/Osborne.

Kessin, Z. (2012). *Programming HTML5 applications*, Beijing ; Farnham: O'Reilly.

Kingsley-Hughes, A.Kingsley-Hughes, K.and Read, D. (2004). *VBScript programmer's reference* 2nd ed., Hoboken, NJ: Wiley.

Korol, J. (2008). *Access 2007 programming by example with VBA, XML, and ASP*, Plano, Tex.: Wordware Pub.

Larsen, R. (2011). An introduction to Ajax. Available at: http://www.ibm.com/developerworks/library/wa-aj-ajaxhistory/ [Accessed February 10, 2017].

Lengstorf, J. (2009). *PHP for absolute beginners*, New York, NY: Distributed by Springer-Verlag New York.

Lupu, R.G.Ungureanu, F.and Siriteanu, V. (2013). Eye tracking mouse for human computer interaction. In *2013 E-Health and Bioengineering Conference (EHB)*. pp. 1–4.

MacDonald, M. (2011). *Html5*, Beijing ; Farnham: O'Reilly.

McClure, W.B. (2007). *Beginning ASP.NET 2.0 AJAX*, Indianapolis, IN: Wrox/Wiley.

McLaughlin, B. (2012). *PHP & MySQL: The Missing Manual*,

Mercer, D. and Shannon, C. (2003). *Html*, New York ; London: McGraw-Hill.

Pepersack, J. (2016). Is Perl good for web development in 2016. *quora*. Available at: https://www.quora.com/Is-Perl-good-for-web-development-in-2016-if-yes-why [Accessed December 15, 2017].

Pinto, A. (2016). Web Page Sizes: A (Not So) Brief History of Page Size through 2015. Available at: http://www.yottaa.com/a-brief-history-of-web-page-size/ [Accessed November 28, 2016].

Pollock, J. (2010). *JavaScript : a beginner's guide* 3rd ed., New York: McGraw Hill.

Powell, T.A. (2008). *Ajax : the complete reference*, New York: McGraw-Hill.

Powell, T.A. (2010). *HTML & CSS : the complete reference* 5th ed., Emeryville, Calif.: McGraw-Hill ; London : McGraw-Hill.

Rouse, M. (2016). Polling. Available at: http://whatis.techtarget.com/definition/polling [Accessed December 10, 2016].

Sarkar, A. (2016). Why Is JavaScript the Programming Language of the Future? *DZone*. Available at: https://dzone.com/articles/why-is-javascript-the-programming-language-of-the [Accessed February 10, 2018].

Satheeq, H. (2015). How do browsers behave when tab or window is not active? *satheeq.blogspot*. Available at: http://satheeq.blogspot.com/2015/12/how-do-browsers-pausechange-javascript.html [Accessed November 30, 2017].

Segue (2013). What is Ajax and Where is it Used in Technology. *Seguetech*. Available at: https://www.seguetech.com/ajax-technology/ [Accessed December 5, 2017].

Sharanam, S. and Vaishali, S. (2008). *Oracle for professionals*, Mumbai: Shroff Publishers & Distributors.

Shelly, G.B.Cashman, T.J.and Woods, D.M. (2002). *HTML complete concepts and techniques* 2nd ed., Cambridge, Mass.: Course Technology.

Shen, L. et al. (2009). A scalable Web GPS monitoring system based on AJAX push pattern. In *Geoinformatics, 2009 17th International Conference*. IEEE, pp. 1–5.

Stackoverflow (2012). Focus-follows-mouse (plus auto-raise) on Mac OS X. Available at: http://stackoverflow.com/questions/98310/focus-follows-mouse-plus-auto-raise-on-mac-os-x [Accessed January 22, 2017].

Stone, D. (2016). Differences Between CSS & CSS3. Available at: https://www.techwalla.com/articles/differences-between-css-css3 [Accessed October 11, 2016].

Sumathi, S. and Esakkirajan, S. (2007). *Fundamentals of relational database management systems*, Berlin ; London: Springer.

Tidwell, D. (2001). *XSLT* 1st ed., Cambridge Mass. .: O'Reilly.

Tobii (2017). This is Eye Tracking. Available at: https://www.tobii.com/group/about/this-is-eye-tracking/ [Accessed January 21, 2017].

W3Schools (2017). SQL Syntax. Available at: https://www.w3schools.com/sql/sql_syntax.asp [Accessed December 16, 2016].

Wang, X. (2011). AJAX technology applications in the network test system. In *Electrical and Control Engineering (ICECE), 2011 International Conference on*. IEEE, pp. 1954–1956.

Wessels, A. et al. (2011). Remote data visualization through websockets. In *Information Technology: New Generations (ITNG), 2011 Eighth International Conference on*. IEEE, pp. 1050–1051.

"What is efficiency" (2018). What is efficiency? Definition and meaning. *MBN*. Available at: https://marketbusinessnews.com/financial-glossary/efficiency-definition-meaning/ [Accessed January 6, 2018].

Widenius, M. and Axmark, D. (2002). *MySQL reference manual : documentation from the source*, Beijing ; Farnham: O'Reilly.

Wikipedia (2016). XSLT. Available at: http://en.wikipedia.org/wiki/Xslt [Accessed December 2, 2016].

Zakas, N.C.McPeak, J.and Fawcett, J. (2007). *Professional Ajax* 2nd ed., Indianapolis, Ind. Chichester: Wiley.

Appendix A The Implementation Code for Browsing Control Model

A.1 Front-End Code for Browsing-Control Implementation

The Listing A-1 below shows the front-end coding of implementing Browsing-Control model upon Polling.

```
<?php

?>
<html>
<head>
<script>
var resultText = "";
var counter1= 0;
var intervalTime = 6000;
var interval_id;
var flag2;

window.addEventListener("blur", function(event)
{  postpone_polling();}, false);

window.addEventListener("focus", function(event)
 { continue_polling();}, false);

function continue_polling()
{
    // To display the updated list according to the interval time.
    interval_id = setInterval(function(){ loadXMLDoc();},intervalTime);
}

function postpone_polling()
{
    clearInterval(interval_id);
}

// This function to show last updates
function loadXMLDoc()
{
    var pollxml;
    if (window.XMLHttpRequest)
    {// code for IE7+, Firefox, Chrome, Opera, Safari
        pollxml=new XMLHttpRequest();
    }
    else
    {// code for IE6, IE5
        pollxml=new ActiveXObject("Microsoft.pollxml");
    }
    pollxml.onreadystatechange=function()
    {
    if (pollxml.readyState==4 && pollxml.status==200)
    {
    counter1 += 1;
    document.getElementById("myDiv2").innerHTML= counter1;
        if(pollxml.responseText != "")
        {
            flag2 = 1;
            if(counter1 < 3)
```

```
                resultText = pollxml.responseText;
            else
                resultText += pollxml.responseText;

            document.getElementById("myDiv").innerHTML= resultText;
        }
    }
    }
    document.getElementById("myDiv3").innerHTML=intervalTime;
    pollxml.open("GET","BrowserControl-Backend.php?flag="+
flag2+"&ct="+counter1 ,true);
    pollxml.send();
}
function AddComment() // To add a member to the list
{
    var xmlhttpList;
    if (window.XMLHttpRequest)
    {// code for IE7+, Firefox, Chrome, Opera, Safari
        xmlhttpList=new XMLHttpRequest();
    }
    else
    {// code for IE6, IE5
        xmlhttpList=new ActiveXObject("Microsoft.XMLHTTP");
    }

    xmlhttpList.onreadystatechange=function()
    {
        if (xmlhttpList.readyState==4 && xmlhttpList.status==200)
        {
            document.getElementById("myDiv4").innerHTML=
xmlhttpList.responseText;
        }
    }
    var str1= document.getElementById("addName").value;
    var str2= document.getElementById("comment").value;
    var newline= "addComment.php?nm=" + str1 + "&com=" + str2 ;
    xmlhttpList.open("GET",newline,true);
    xmlhttpList.send();
}
</script>
</head>
<body>
<p style="display: inline;">Enter your name   <input type="text" id=
"addName" value="">
<div style="display: inline; color:red;">* only at first time!</div>
</p>
<p style="display:
inline;">         Type your
comment here:   <input type="text" id= "comment" value="">
   <button id="btn" type="button"
onclick="AddComment()">Submit</button>  <div id="myDiv4"
style="display: inline; color:red;"></div>
</p>
<p align="center">--------------- Comments board ----------------</p>
<div id="myDiv"><h2></h2></div>
<br>
<p style="display: inline; padding-left:80px;"> Current interval == :
<div id="myDiv3" style="display: inline; color:red;"></div> ms</p>

<p style="display: inline; padding-left:80px;"> Number of attempts  == :
<div id="myDiv2" style="display: inline; color:red;"><h2></h2></div></p>
</body>
</html>
```

Listing A-1 Browsing-Control model: front-end code.

A.2 Back-end Code for Browsing-Control Implementation

The Listing A-2 below shows the back-end coding of implementing Browsing-Control model.

```
<?php
//connect to database
$link = mysql_connect('localhost', 'root')
  or die('Could not connect: ' . mysql_error());
$dbcon  =  mysql_select_db('hatem',$link)  or  die('Could  not  select
database');

//execute query
SESSION_START();

if(isset($_GET['flag']))
   if($_GET['flag'] == 0)
      $_SESSION['recordN'] = 0 ;

$id = $_SESSION['recordN'];

$attempts = $_GET['ct'];
$sql = "SELECT * FROM chat where SN >" .$id ;
$result = mysql_query($sql);
$nRows = mysql_num_rows($result);
date_default_timezone_set('Europe/London');
if($nRows > 0)
{
    $dt = new DateTime();
    $currentTime = $dt->format('H:i:s');
    echo "<table col=5 border=1 align= center style='padding: 0px; border:
solid #000 1px;'>";
    if($id == 0)
    {
      echo "<tr bgcolor=#DADADA><td width=120 align=middle>User</td><td
width=300 align=middle> Comment</td>
      <td   width=100   align=middle>Insert   time</td><td   width=100
align=middle>Current                time</td><td                width=60
align=middle>Attempts</td></tr>";
    }

    while ($row = mysql_fetch_array($result))
    {
        $_SESSION['recordN'] = $row['SN'];
        $cn = $_SESSION['recordN'];
        $dt = new DateTime();
        $currentTime = $dt->format('H:i:s');
        echo   "<tr><td   align=middle   width=120>$row[userName]</td><td
align=middle width=300>$row[comment]</td>";
       echo   "<td   align=middle   width=100>$row[additionTime]</td><td
align=middle         width=100>$currentTime</td><td         align=middle
width=60>$attempts</td></tr>";
    }
    echo "</table>";
}
mysql_close($link);
?>
```

Listing A-2 Browsing-Control model: bac-kend code.

Appendix B The implementation code for Polling technique.

B.1 Polling Implementation Front-End Code

The Listing B-1 below shows the front-end coding of a simple chat webpage for Polling implementation.

```
<?php
SESSION_START();
$_SESSION['recordN']= 0;
?>
<html>
<head>
<script>

var resultText = "";
var counter1= 0;
var intervalTime = 1000;
var puls;
var flag2;

function loadXMLDoc() // This function to refresh the list frequently,
according to interval time!!
{
    var pollxml;
    if (window.XMLHttpRequest)
    {// code for IE7+, Firefox, Chrome, Opera, Safari
        pollxml=new XMLHttpRequest();
    }
    else
    {// code for IE6, IE5
        pollxml=new ActiveXObject("Microsoft.pollxml");
    }
    pollxml.onreadystatechange=function()
    {
      if (pollxml.readyState==4 && pollxml.status==200)
      {
      counter1 += 1;
      document.getElementById("myDiv2").innerHTML= counter1;
        if(pollxml.responseText != "")
        {
            flag2 = 1;
            if(counter1 < 3)
                resultText = pollxml.responseText;
            else
                resultText += pollxml.responseText;

            document.getElementById("myDiv").innerHTML= resultText;
        }
      }
    }
    document.getElementById("myDiv3").innerHTML=intervalTime;
    pollxml.open("GET","pollScript.php?flag="+      flag2+"&ct="+counter1
,true);
    pollxml.send();
}

// To display the updated list
```

```
var puls = setInterval(function(){ loadXMLDoc();},intervalTime);
</script>
</head>
<body>
<div id="myDiv"><h2></h2></div>
<br>
<p style="display: inline; padding-left:80px;"> Current interval == : <div
id="myDiv3" style="display: inline; color:red;"></div> ms</p>

<p style="display: inline; padding-left:80px;"> Number of attempts  == :
<div id="myDiv2" style="display: inline; color:red;"><h2></h2></div></p>

</body>
</html>
```

Listing B-1 Polling implementation: front-end code.

B.2 Polling Implementation Back-End Code

The Listing B-2 below shows the back-end coding of a simple chat webpage for Polling implementation.

```
<?php
//connect to database
$link = mysql_connect('localhost', 'root')
  or die('Could not connect: ' . mysql_error());
$dbcon  =  mysql_select_db('hatem',$link)  or  die('Could  not  select
database');

//execute query
SESSION_START();

if(isset($_GET['flag']))
   if($_GET['flag'] == 0)
      $_SESSION['recordN'] = 0 ;

$id = $_SESSION['recordN'];

$attempts = $_GET['ct'];
$sql = "SELECT * FROM chat where SN >" .$id ;
$result = mysql_query($sql);
$nRows = mysql_num_rows($result);
date_default_timezone_set('Europe/London');
if($nRows > 0)
{
    $dt = new DateTime();
    $currentTime = $dt->format('H:i:s');
    echo "<table col=5 border=1 align= center style='padding: 0px; border:
solid #000 1px;'>";
    if($id == 0)
    {
      echo  "<tr  bgcolor=#DADADA><td  width=120  align=middle>User</td><td
width=300 align=middle> Comment</td>
      <td    width=100    align=middle>Insert    time</td><td    width=100
align=middle>Current                time</td><td                width=60
align=middle>Attempts</td></tr>";
    }

    while ($row = mysql_fetch_array($result))
    {
```

```
        $_SESSION['recordN'] = $row['SN'];
        $cn = $_SESSION['recordN'];
        $dt = new DateTime();
        $currentTime = $dt->format('H:i:s');
        echo   "<tr><td   align=middle   width=120>$row[userName]</td><td
align=middle width=300>$row[comment]</td>";
        echo   "<td   align=middle   width=100>$row[additionTime]</td><td
align=middle          width=100>$currentTime</td><td          align=middle
width=60>$attempts</td></tr>";
    }
    echo "</table>";
}
mysql_close($link);
?>
```

Listing B-2 Polling implementation: back-end code

Appendix C The Implementation Code for Adaptive Polling.

C.1 Adaptive Polling Implementation Front-End Code

The Listing C-1 below shows the front-end coding of a simple chat webpage for Adaptive Polling implementation.

```
<? php
SESSION_START();
$_SESSION['recordNumber']= 0;

?>
<html>
<head>
<script>

var resultText = "";
var counter1= 0;
var minInterval = 1000
var maxInterval= 6000;
var intervalValue= 500;
var intervalTime = minInterval;
var flag2= 0;
var si;

function loadXMLDoc() // This function to refresh the list frequently, depending on the speed of updates!!
{
    var xmlhttp;
    if (window.XMLHttpRequest)
    {// code for IE7+, Firefox, Chrome, Opera, Safari
        xmlhttp=new XMLHttpRequest();
    }
    else
    {// code for IE6, IE5
        xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
    }
    xmlhttp.onreadystatechange=function()
    {
    if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {

    counter1 += 1;
    document.getElementById("myDiv2").innerHTML= counter1;

        if(xmlhttp.responseText != "")
        {
            flag2 = 1;
            if(counter1 < 3)
                resultText = xmlhttp.responseText;
            else
                resultText += xmlhttp.responseText;


            document.getElementById("myDiv").innerHTML= resultText;
            changeInterval(false);
        }
```

```
        else
        {
           changeInterval(true);
        }
    }
    }
    document.getElementById("myDiv3").innerHTML=intervalTime;
    xmlhttp.open("GET","myscript.php?flag="+        flag2+"&ct="+counter1
,true);
    xmlhttp.send();
}

// To display the updated list.
var si = setInterval(function(){ loadXMLDoc();},intervalTime);

// To increase or decrease the interval time.
function changeInterval(flag)
{
    if (!flag)
    {
        if (intervalTime > minInterval)
        {
            intervalTime -= intervalValue;
            clearInterval(si);
            si = setInterval(function(){

                  loadXMLDoc();},intervalTime);
        }
    }
    else
    {
        if (intervalTime < maxInterval)
        {
            intervalTime += intervalValue;
            clearInterval(si);
            si = setInterval(function(){loadXMLDoc();},intervalTime);

        }
    }
}

</script>
</head>
<body>

<div id="myDiv"><h2></h2></div>
<br>
<p style="display: inline; padding-left:80px;"> Current interval == : <div
id="myDiv3" style="display: inline; color:red;"></div> ms</p>

<p style="display: inline; padding-left:80px;"> Number of attempts  == :
<div id="myDiv2" style="display: inline; color:red;"><h2></h2></div></p>

</body>
</html>
```

Listing C-1 Adaptive Polling implementation: front-end code

C.2 Adaptive Polling Implementation Back-End Code

The Listing C-2 shows the back-end coding of a simple chat webpage for Adaptive Polling implementation.

```
<?php
//connect to database
$link = mysql_connect('localhost', 'root')
  or die('Could not connect: ' . mysql_error());
$dbcon   =   mysql_select_db('hatem',$link)   or   die('Could   not   select
database');


//execute query
SESSION_START();

if(isset($_GET['flag']))
   if($_GET['flag'] == 0)
      $_SESSION['recordNumber'] = 0 ;

$id = $_SESSION['recordNumber'];

$attempts = $_GET['ct'];
$sql = "SELECT * FROM chat where SN >" .$id ;
$result = mysql_query($sql);
$nRows = mysql_num_rows($result);
date_default_timezone_set('Europe/London');
if($nRows > 0)
{
    $dt      = new DateTime();
    $currentTime = $dt->format('H:i:s');
    echo "<table col=5 border=1 align= center style='padding: 0px; border:
solid #000 1px;'>";
    if($id == 0)
    {
      echo  "<tr  bgcolor=#DADADA><td  width=120  align=middle>User</td><td
width=300 align=middle> Comment</td>
      <td    width=100    align=middle>Insert    time</td><td    width=100
align=middle>Current                time</td><td                width=60
align=middle>Attempts</td></tr>";
    }

    while ($row = mysql_fetch_array($result))
    {
        $_SESSION['recordNumber'] = $row['SN'];
        $cn = $_SESSION['recordNumber'];
        $dt      = new DateTime();
        $currentTime = $dt->format('H:i:s');
        echo    "<tr><td    align=middle    width=120>$row[userName]</td><td
align=middle width=300>$row[comment]</td>";
       echo    "<td    align=middle    width=100>$row[additionTime]</td><td
align=middle          width=100>$currentTime</td><td          align=middle
width=60>$attempts</td></tr>";
    }
    echo "</table>";
}

mysql_close($link);
?>
```

Listing C-2 Adaptive Polling implementation: back-end code.

C.3 Add-Comments Function Code for Adaptive Polling Implementation.

The Listing C-3 below shows the back-end coding of adding new comments to chat webpage of Adaptive Polling.

```
<?php
//connect to database
$link = mysql_connect('localhost', 'root')
  or die('Could not connect: ' . mysql_error());
mysql_select_db('hatem') or die('Could not select database');

if (isset ($_GET["nm"]))
{
    $user = preg_replace('/[^A-Za-z0-9\. -]/', ' ', $_GET["nm"]);
    $comment= preg_replace('/[^A-Za-z0-9\. -]/', ' ',$_GET["com"]);
    if (($user !== '')and($comment !== ''))
    {
    date_default_timezone_set('Europe/London');
    $dt = new DateTime();
    $currentTime = $dt->format('H:i:s');
    $sql     =     "Insert     into     chat(userName,comment,additionTime)
values('$user','$comment','$currentTime')";
    if(mysql_query($sql))
        echo "<font color=red>Sent</font>";
    else
        echo "<font color=red>Failed</font>";
    }
}
mysql_close($link);
?>
```

Listing C-3 Add-Comments for Adaptive Polling: back-end code.

Appendix D The Code and Screenshot for Adaptive Polling Vs Polling Simulation.

D.1 The Simulation Front-End Code

The Listing D-1 below shows the front-end coding for a comparison simulation between Adaptive Polling and Polling.

```
<?php

?>
<html>
<head>
<script>

function AddComment() // To add a comment to the list
{
    var xmlhttpList;
    if (window.XMLHttpRequest)
    {// code for IE7+, Firefox, Chrome, Opera, Safari
        xmlhttpList=new XMLHttpRequest();
    }
    else
    {// code for IE6, IE5
        xmlhttpList=new ActiveXObject("Microsoft.XMLHTTP");
    }
    xmlhttpList.onreadystatechange=function()
    {
        if (xmlhttpList.readyState==4 && xmlhttpList.status==200)
        {
            document.getElementById("myDiv4").innerHTML=
            xmlhttpList.responseText;
        }
    }
    var str1= document.getElementById("addName").value;
    var str2= document.getElementById("comment").value;
    var newline= "addComment.php?nm=" + str1 + "&com=" + str2 ;
    xmlhttpList.open("GET",newline,true);
    xmlhttpList.send();
}
</script>
</head>
<body>
</br>

<p style="display: inline;">Enter your name  
<input type="text" id= "addName" value="">
<div style="display: inline; color:red;">* only at first time!</div>
<p  style="display: inline;">      Type your
comment here:  
<input type="text" id= "comment" value="">
```

```
<button id="btn" type="button" onclick="AddComment()">Submit</button>
  <div id="myDiv4" style="display: inline; color:red;"></div>
</p>
<p align="center">---------------- Comments board ----------------</p>

<table width="98%" cols="2" border="0">
<tr align="center">
<td width="49%">Adaptive Polling</td>
<td width="49%">Normal Polling</td>
</tr>
</table>

<iframe name= "adaptivePolling" width="49%" src="adaptivePolling.php"
height="1000" frameborder="0">
</iframe>

<iframe name= "polling" width="49%" src="polling.php" height="1000"
frameborder="0">
</iframe>

</body>
</html>
```

Listing D-1 Coding for Adaptive Polling and Polling simulation.

D.2 The Simulation Screenshot

The Figure D-1 shows a screenshot of the comparison simulation between Adaptive Polling and Polling.

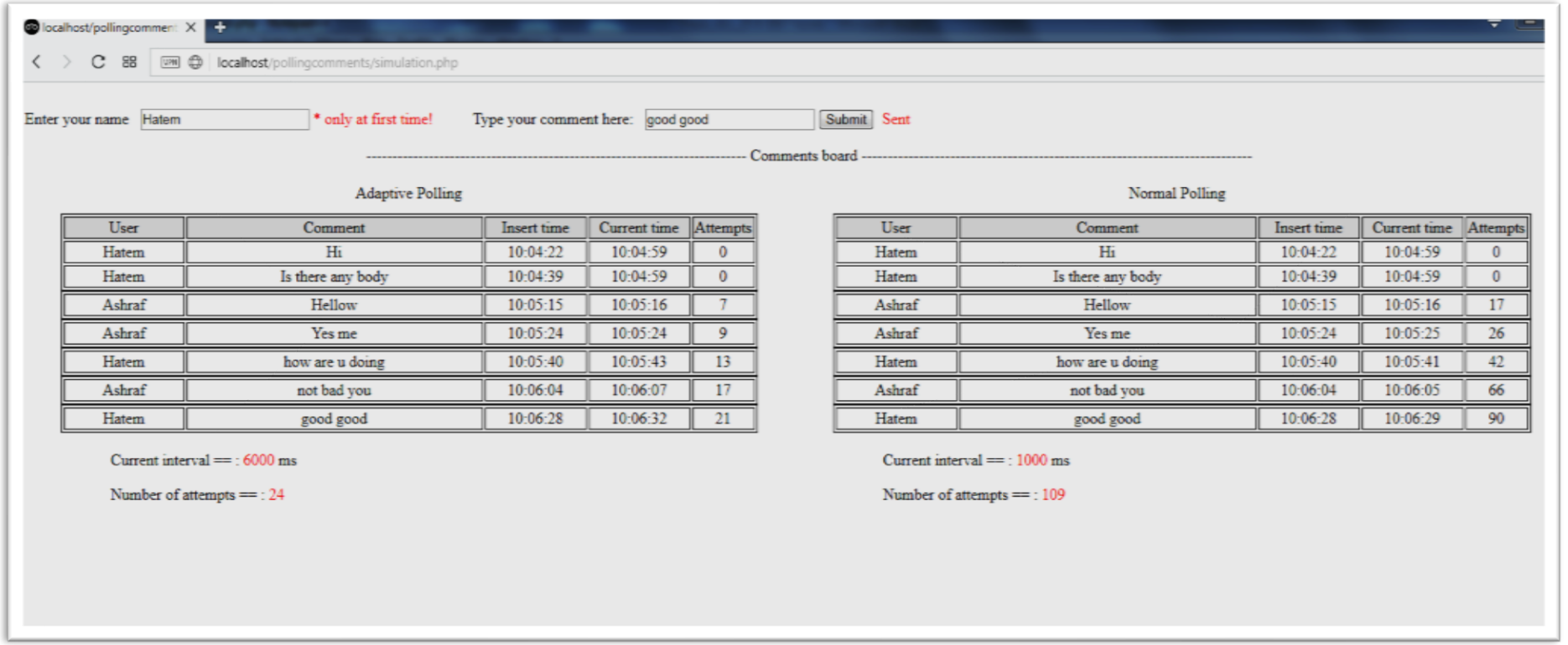

Figure D-1 Screenshot of comparison simulation between Adaptive Polling and Polling.

Appendix E: Publications and Presentations

Conferences:

- Aziz, H. and Ridley, M. Real-time web applications driven by active browsing. In Internet Technologies and Applications (ITA) 2017, pp. 4-8. IEEE, Wrexham,
- Aziz, H. and Ridley, M. Adaptive Polling for Responsive Web Applications. In Information Science and Applications (ICISA) 2016 (pp. 1157-1167). Springer, Singapore.

Seminars:

- “Enhanced Polling for Real-Time Websites” as a seminar presented at University of Bradford ACM ‘Association for Computer Machinery’, Bradford, 19th of May 2016.